

2100 0410 #4
PATENT APPLICATION

IN THE UNITED STATES PATENT AND TRADEMARK OFFICE

In re the Application of

Kenichi NUMATA et al.

Application No.: 10/014,661

Filed: December 14, 2001

Docket No.: 111470

For: STRUCTURED DOCUMENT MANAGEMENT SYSTEM, STRUCTURED DOCUMENT
MANAGEMENT METHOD, SEARCH DEVICE AND SEARCH METHOD

CLAIM FOR PRIORITY

RECEIVED

Director of the U.S. Patent and Trademark Office
Washington, D.C. 20231

FEB 21 2002

Technology Center 2100

Sir:

The benefit of the filing date of the following prior foreign application filed in the following foreign country is hereby requested for the above-identified patent application and the priority provided in 35 U.S.C. §119 is hereby claimed:

Japanese Patent Application No. 2001-255016 filed August 24, 2001.

In support of this claim, a certified copy of said original foreign application:

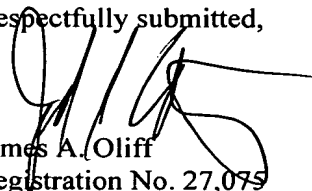
 X is filed herewith.

 was filed on in Parent Application No. filed .

 will be filed at a later date.

It is requested that the file of this application be marked to indicate that the requirements of 35 U.S.C. §119 have been fulfilled and that the Patent and Trademark Office kindly acknowledge receipt of this document.

Respectfully submitted,


James A. Oliff
Registration No. 27,079

Joel S. Armstrong
Registration No. 36,430

JAO:JSA/zmc
Date: February 1, 2002

OLIFF & BERRIDGE, PLC
P.O. Box 19928
Alexandria, Virginia 22320
Telephone: (703) 836-6400

DEPOSIT ACCOUNT USE
AUTHORIZATION
Please grant any extension
necessary for entry;
Charge any fee due to our
Deposit Account No. 15-0461



日 本 国 特 許 庁
JAPAN PATENT OFFICE

別紙添付の書類に記載されている事項は下記の出願書類に記載されている事項と同一であることを証明する。

This is to certify that the annexed is a true copy of the following application as filed with this Office

出 願 年 月 日

Date of Application:

2001年 8月24日

出 願 番 号

Application Number:

特願2001-255016

出 願 人

Applicant(s):

富士ゼロックス株式会社

RECEIVED

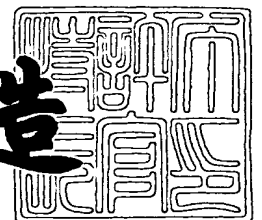
FEB 21 2002

Technology Center 2100

2001年12月 7日

特 許 庁 長 官
Commissioner,
Japan Patent Office

及 川 耕 造



出証番号 出証特2001-3106893

【書類名】 特許願

【整理番号】 FE01-01151

【提出日】 平成13年 8月24日

【あて先】 特許庁長官 殿

【国際特許分類】 G06F 17/30

【発明者】

 【住所又は居所】 神奈川県横浜市西区みなとみらい3丁目3番1 富士ゼ
 ロックス株式会社内

 【氏名】 沼田 賢一

【発明者】

 【住所又は居所】 神奈川県横浜市西区みなとみらい3丁目3番1 富士ゼ
 ロックス株式会社内

 【氏名】 川邊 恵久

【発明者】

 【住所又は居所】 神奈川県横浜市西区みなとみらい3丁目3番1 富士ゼ
 ロックス株式会社内

 【氏名】 額賀 雅夫

【発明者】

 【住所又は居所】 神奈川県横浜市西区みなとみらい3丁目3番1 富士ゼ
 ロックス株式会社内

 【氏名】 山田 季史

【発明者】

 【住所又は居所】 神奈川県横浜市西区みなとみらい3丁目3番1 富士ゼ
 ロックス株式会社内

 【氏名】 池田 稔

【発明者】

 【住所又は居所】 神奈川県横浜市西区みなとみらい3丁目3番1 富士ゼ
 ロックス株式会社内

 【氏名】 東 和彦

【発明者】

【住所又は居所】 神奈川県横浜市西区みなとみらい3丁目3番1 富士ゼ
ロックス株式会社内

【氏名】 山田 美穂

【特許出願人】

【識別番号】 000005496

【氏名又は名称】 富士ゼロックス株式会社

【代理人】

【識別番号】 100101948

【弁理士】

【氏名又は名称】 柳澤 正夫

【電話番号】 (045)744-1878

【手数料の表示】

【予納台帳番号】 059086

【納付金額】 21,000円

【提出物件の目録】

【物件名】 明細書 1

【物件名】 図面 1

【物件名】 要約書 1

【包括委任状番号】 9204691

【プルーフの要否】 要

【請求項 5】 前記構造検索手段は、2つの要素が同一の部分構造に含まれているときには前記第 2 の構造情報を用いて先祖・子孫関係を求めることを特徴とする請求項 3 または請求項 4 に記載の検索装置。

【請求項 6】 構造化文書中の要素間の先祖・子孫関係を検索条件とした構造検索を行う検索方法において、構造化文書を設定に従って分解した複数の部分構造間の関係を示す第 1 の構造情報と前記部分構造ごとに該部分構造中の各要素間の関係を示す第 2 の構造情報を保持しておき、それぞれの要素が含まれる部分構造間の先祖・子孫関係を前記第 1 の構造情報によって求めるとともに、部分構造が先祖・子孫関係にあるときは、さらに先祖側の部分構造から子孫側の部分構造へのパス上にあつて前記先祖側の部分構造の子の部分構造のルートとなる要素と前記先祖側の部分構造に含まれる要素との間の先祖・子孫関係を前記第 2 の構造情報によって求めることを特徴とする検索方法。

【請求項 7】 前記先祖側の部分構造に含まれる要素が当該先祖側の部分構造のルートとなる要素である場合には、前記第 2 の構造情報を用いた判定を行わないことを特徴とする請求項 6 に記載の検索方法。

【請求項 8】 2つの要素が同一の部分構造に含まれているときには前記第 2 の構造情報を用いて先祖・子孫関係を求めることを特徴とする請求項 6 または請求項 7 に記載の検索方法。

【発明の詳細な説明】

【0001】

【発明の属する技術分野】

本発明は、大規模な構造化文書を編集可能に管理する構造化文書管理装置及び構造化文書管理方法と、管理されている構造化文書中の要素間の先祖・子孫関係を検索する検索装置及び検索方法に関するものである。

【0002】

【従来の技術】

従来より、XML などの構造化文書をデータベース化し、内容や文書構造での検索や、部分的な再利用などの利用に供している。一般に数ページ程度の小さな文書では、編集者が 1 人で文書の作成、編集を行う場合が多く、構造化文書をデ

ータベース化した場合でも、文書全体を編集対象として編集作業が行われる場合が多い。しかし、大規模な文書では複数人が文書中のそれぞれの担当部分を編集してゆくといったことが行われており、構造化文書をデータベース化した場合でも、このような共同編集環境の提供が必要となる。

【0003】

従来、構造化文書をデータベース化する場合には、例えば文書中の各要素の親子関係を示すポインタなどで結んで文書中の構造を保持したり、構造を示すインデックステーブルなどを用いて構造を保持している。他にもいくつかの方法が考えられているが、従来の方法はいずれも、文書全体を一律に画一的なデータ形式によって保持している。

【0004】

このような従来の構造化文書の格納方法を用いている場合、編集した文書を格納すると、その影響が文書全体に波及してしまうことも少なくない。そのため、上述のような共同編集環境を考えた場合、従来の方法では複数人がそれぞれ並行して編集することができなかった。

【0005】

また、編集により文書全体に影響が及ぶ場合、文書に対して編集が加えられる度に、文書全体のデータを再構築する必要が発生する。大規模な文書では、このようなデータの再構築には多大な処理時間がかかり、編集するたびにデータの再構築を行っていたのでは非常に効率が悪いという問題があった。

【0006】

一方、このように構造化文書をデータベース化した場合、文書構造の検索、特に2つの要素間の先祖・子孫関係の検索が頻繁に行われる。先祖・子孫関係は、ある要素から上位層へ要素をたぐってゆくだけで、あるいは下位層へ要素をたぐってゆくだけで、たどり着くことができる要素間の関係を示す。2つの要素が上下関係で直接接続されている関係が親子関係であるが、先祖・子孫関係は親子関係を含むものである。

【0007】

このような先祖・子孫関係の検索は、頻繁に行われることから高速に実行でき

ることが望まれる。従来は要素を1つずつたぐってゆくことによって行うが、枝分かれが多く、また階層が深い文書では、要素の探索時間も長くなってしまう。そのため、大規模な文書において先祖・子孫関係の検索を効率よく行うことができる手法が要望されていた。

【0008】

【発明が解決しようとする課題】

本発明は、上述した事情に鑑みてなされたもので、大規模な構造化文書を効率よく取り扱うことができるとともに、複数人が共同して編集可能な環境を提供できる構造化文書管理装置及び構造化文書管理方法を提供することを目的とするものである。さらに、このような構造化文書管理装置及び構造化文書管理方法により管理されている構造化文書中の要素の先祖・子孫関係を効率よく判定して文書構造の検索を高速に行うことができる検索装置及び検索方法を提供することを目的とするものである。

【0009】

【課題を解決するための手段】

本発明は、構造化文書を管理する構造化文書管理装置及び構造化文書管理方法において、入力された構造化文書を設定に従って複数の部分構造に分解するとともに、部分構造間の関係を第1の構造情報として生成する。さらに、分解された部分構造ごとに、部分構造中の各要素間の関係を第2の構造情報として生成する。そして、第1の構造情報と第2の構造情報を保持して、構造化文書を管理する。このように、構造化文書を部分構造に分解して、その構造を第1及び第2の構造情報として保持するので、部分構造単位での編集を行うことによって、編集による影響を部分構造内に収めることができる。従って、複数人がそれぞれ担当する部分構造を並行して編集する場合でも、他の部分構造への影響をなくし、共同編集環境を提供することができる。また、構造が変更された場合でも、そのためのデータの再構築は部分構造内で行えばよいので、高速に再構築のための処理を行うことができる。

【0010】

また本発明は、このように構造化文書を設定に従って分解した複数の部分構造

間の関係を示す第1の構造情報と前記部分構造ごとに該部分構造中の各要素間の関係を示す第2の構造情報を保持している場合に、構造化文書中の要素間の先祖・子孫関係を検索条件とした構造検索を行う検索装置及び検索方法であって、まず、それぞれの要素が含まれる部分構造間の先祖・子孫関係を第1の構造情報によって求める。部分構造が先祖・子孫関係にあるときは、さらに先祖側の部分構造から子孫側の部分構造へのパス上にあつて先祖側の部分構造の子の部分構造のルートとなる要素と、先祖側の部分構造に含まれる要素との間の先祖・子孫関係を、第2の構造情報によって求める。これによって、従来のようにそれぞれの要素を1つずつたぐってゆかなくても、2つの要素が先祖・子孫関係にあるか否かを高速に求めることができ、大規模な構造化文書の検索を効率的に行うことができる。

【0011】

なお、この検索を行う際に、先祖側の部分構造に含まれる要素が当該先祖側の部分構造のルートとなる要素である場合には、前記第2の構造情報を用いた判定を行わず、第1の構造情報を用いた処理のみで2つの要素が先祖・子孫関係にあるか否かを求めることができる。また、2つの要素が同一の部分構造に含まれているときには、第1の構造情報を用いることなく、第2の構造情報を用いて先祖・子孫関係を求めることができる。

【0012】

【発明の実施の形態】

図1は、本発明の実施の一形態を示すブロック図である。図中、1は文書入力部、2はファイル管理部、3はディスパッチャ、4は検索部、5は文書処理部、6は文書解析部、7は文書分解部、8は編集単位処理部、9は文書解析部、10は差分解部、11は要素情報登録部、12は構造情報登録部、13はリレーショナルデータベースである。文書入力部1は、構造化文書をファイル管理部2に対して入力する。このとき、ここでは分解前の構造化文書であるか、分解された部分構造であるかを示すコンテンツタイプも入力するものとする。また文書入力部1は、ファイル管理部2の検索部4に対して構造化文書あるいは構造化文書の部分構造の取り出しを要求して部分構造を受け取る。受け取った構造化文書ある

いはその部分構造に対して編集を行った結果を再びファイル管理部2に対して入力することができる。また、構造化文書に対する文書構造の検索などを要求することができる。

【0013】

ファイル管理部2は、文書入力部1との間で構造化文書あるいはその部分構造の受け渡しを行う。ファイル管理部2は、ディスパッチャ3及び検索部4を有している。ディスパッチャ3は、文書入力部1から渡される構造化文書あるいはその部分構造を受け取り、そのコンテンツタイプに従って、部分構造への分解前の構造化文書の場合には、その構造化文書を文書処理部5に渡す。また、分解された部分構造を受け取った場合には、その部分構造を編集単位処理部8へ渡す。さらに、文書処理部5で分解された部分構造を受け取って編集単位処理部8へ渡す。

【0014】

検索部4は、文書入力部1から要求された構造化文書あるいはその部分構造をリレーショナルデータベース13から取り出し、文書入力部1へ渡す。また、各種のリレーショナルデータベース13に対する検索要求を受け付け、検索結果を返す。特に文書構造の検索を行うことができ、このとき、本発明の検索方法による要素間の先祖・子孫関係の判定を行う。

【0015】

文書処理部5は、ファイル管理部2のディスパッチャ3から渡される構造化文書を解析して、部分構造に分解する。文書処理部5は、構造化文書を解析する文書解析部6と、リレーショナルデータベース13から読み込んだ設定に従って構造化文書を複数の部分構造に分解する文書分解部7を有している。大規模な構造化文書では、分解された部分構造が編集単位となる。分解された部分構造には部分構造であることを示すコンテンツタイプが付加され、ファイル管理部2に入力される。また、部分構造間の関係をグローバル構造情報（第1の構造情報）として、リレーショナルデータベース13にレコードとして登録する。

【0016】

編集単位処理部8は、ファイル管理部2から渡される部分構造について、部分

構造内の要素の情報を要素情報として、また各要素間の関係を構造情報として、それぞれリレーショナルデータベース13に格納する。編集単位処理部8は、文書解析部9、差分析部10、要素情報登録部11、構造情報登録部12を有している。文書解析部9は、ファイル管理部2から渡された部分構造を解析する。

【0017】

この例では、編集された部分構造については編集前の部分構造との差分をリレーショナルデータベース13に格納することにより版管理の利用に供するものとし、そのための構成として差分析部10を有している。差分析部10は、文書解析部9による解析結果に従い、リレーショナルデータベース13に保持されている編集前の部分構造と編集後の部分構造の差分を抽出してリレーショナルデータベース13への登録対象とする。なお、新規の構造化文書の部分構造である場合には、部分構造全体をリレーショナルデータベース13への登録対象とする。差分管理を行わない場合には、この差分析部9は不要である。

【0018】

要素情報登録部11は、部分構造中の各要素について、それぞれの属性情報や内容をフィールド値とするレコードをリレーショナルデータベース13に登録する。なお、差分析部10による差分の抽出で追加されたことが判明した要素については新規の登録を行い、変更及び削除されたことが判明した要素については削除処理を行う。削除処理は、実際にレコードを削除してもよいし、例えばバージョン管理を行う場合には、削除された旨の情報を付加して残しておいてもよい。

【0019】

構造情報登録部12は、ファイル管理部2から渡された部分構造について、各要素間の関係を構造情報(第2の構造情報)として生成し、リレーショナルデータベース13に登録する。部分構造に対して編集が行われた場合には、その部分構造の構造情報を再構築して新たに登録する。構造情報は、このように編集が行われると再構築が必要となるが、分解された部分構造ごとに行うので、他の部分構造には影響を与えずに更新することが可能であり、文書全体の文書構造の再構築を行う必要はない。なお、バージョン管理を行う際には、当該部分構造が編集

される都度、構造情報を再構築してグローバル構造情報との対応付けを行っておけばよい。

【0020】

リレーショナルデータベース13は、一般的なりレーショナルデータベースであり、構造化文書の各要素の要素情報と、部分構造間の関係を示すグローバル構造情報と、部分構造内の各要素間の関係を示す構造情報をそれぞれレコードとして格納し、これらの情報によって構造化文書を保持する。なお、グローバル構造情報及び構造情報については、その検索性能を高めるため、バイナリ形式のデータとしてレコード中のフィールドに格納しておくもよい。もちろんこれらのデータを別のデータベースに登録しておいてもよい。

【0021】

次に、上述の本発明の実施の一形態における概括的な動作の一例を説明する。図2は、構造化文書の構造の一例の説明図、図3は、グローバル構造情報及び要素情報と構造情報の一例の説明図である。ここでは図2(A)に示すように5階層の構造化文書が入力される場合を考える。各要素を○印で示しており、数字は各要素を特定するシーケンス番号(SNo)である。

【0022】

文書入力部1から図2(A)に示すような構造化文書(及びコンテンツタイプ)が入力されると、ディスパッチャ3は、入力された構造化文書のコンテンツタイプに応じ、ここでは文書処理部5へ入力された構造化文書を渡す。

【0023】

文書処理部5では、文書解析部6で構造化文書を解析し、文書分解部7でリレーショナルデータベース13から読み込んだ設定に従い、文書を複数の部分構造に分解する。例えば図2(A)において三角形で示した部分ごとに構造化文書を分解する。これによって図2(B)に示すように3つの部分構造となる。分解された部分構造は、再びファイル管理部2のディスパッチャ3に渡される。

【0024】

このように部分構造への分解を行ったときに、各部分構造の関係を示すグローバル構造情報を生成してリレーショナルデータベース13に格納する。図2(B

）に示すように分解した3つの部分構造にそれぞれグローバルID（GIDと略す）として0, 1, 2と振り、構造を示す情報として、説明を簡単にするため親のGIDを保持するとすれば、図3（A）に示すようなそれぞれの部分構造の親のGIDを並べたデータが得られる。このようなデータを、例えばメモリモージとしてそのままレコード中のフィールドとして埋め込み、ここでは文書を一意に特定するための文書IDとともにリレーショナルデータベース13に格納する。

【0025】

なお、部分構造へ分解する際に、下位の部分構造の頂点の要素が、その上位の部分構造にも含まれることになる。例えば図2に示した例では、SNoが3, 8の要素については複数の部分構造にまたがる。このような場合には、上位の部分構造にダミーの要素を付加しておく。この例ではGIDが0の部分構造に、SNoが3の要素の代わりにSNoが13の要素を、SNoが8の要素の代わりにSNoが14の要素を仮想的に付加している。なお、付加したダミーの要素を、マウントポイントと呼ぶことにする。

【0026】

文書分解部7で分解された部分構造を受け取ったディスパッチャ3では、今度は部分構造を編集単位処理部8に渡す。編集単位処理部8では、文書解析部9で部分構造を解析し、差解析部10に渡す。ここでは新規の構造化文書であるので、差解析部10は部分構造を要素情報登録部11及び構造情報登録部12に渡す。

【0027】

要素情報登録部11では、部分構造中のそれぞれの要素に関する情報を要素情報としてリレーショナルデータベース13に登録する。例えば図2（B）に示したGIDが2の部分構造の場合、SNoが8～12の各要素の情報をリレーショナルデータベース13に登録することになる。図3（B）に要素情報の一例を示しており、この例では各要素を特定するSNoとともに、部分構造中で要素を特定するローカルナンバ（LNo）、要素名、属性値をフィールド値としている。もちろん、フィールドの構成は任意であり、このほかにバージョン情報などを含

んでいてもよい。この要素情報では各要素間の親子関係などは含まれていないが、例えば属性値の検索など、フィールド値による高速な検索が可能である。また、後述する構造情報とは、例えばLN○やSN○によって対応づけることができる。

【0028】

構造情報登録部12では、部分構造中の各要素間の関係を示す構造情報を生成する。構造情報は、どのようなデータ構造であってもよいが、図3(C)に示す例では、それぞれの要素の親のLN○を並べたデータとして構成している。また、この例では構造情報のメモリイメージをそのままレコード中のフィールドとして埋め込み、さらに部分構造を一意に特定するためのGIDとともにリレーショナルデータベース13に格納する。

【0029】

なお、図3(B)、(C)には、GIDが2の部分構造における要素情報及び構造情報のみを示しているが、同様の要素情報及び構造情報が、GIDが0, 1の部分構造についても作成され、リレーショナルデータベース13に登録される。また、図3に示したグローバル構造情報、要素情報、構造情報のデータ構造は一例であり、リレーショナルデータベース13の設計時に任意に取り決めておくことができる。例えば後述する具体例においては、図3に示したデータ構造とは一部異なるデータ構造を用いている。

【0030】

上述のようにして構造化文書が分解されてリレーショナルデータベース13に登録されていると、部分構造を編集単位として文書の編集を行うことができる。例えば文書入力部1から部分構造の取り出しをファイル管理部2に要求することによって、検索部4が編集すべき部分構造をリレーショナルデータベース13から読み出して文書入力部1に対して出力する。そして、文書入力部1において部分構造に対して編集処理を行った後、再びファイル管理部2に入力する。

【0031】

ファイル管理部2に入力された編集後の部分構造（及びコンテンツタイプ）は、ディスパッチャ3によって編集単位処理部8へ送られる。編集単位処理部8で

は、文書解析部 9 で部分構造を解析し、差分解析部 10 に渡す。差分解析部 10 では、リレーショナルデータベース 13 に登録されている編集前の部分構造と文書解析部 9 から受け取った編集後の部分構造を比較し、その差分を抽出する。要素の編集としては、内容や属性値の変更の他、新たな追加、あるいは要素の削除などがある。要素情報登録部 11 では、新たに追加された要素については、要素情報をリレーショナルデータベース 13 に登録する。このとき、追加時のバージョン情報を付加してもよい。また、削除された要素については、レコードを削除したり、あるいは、当該要素に対応する要素情報に削除された旨の情報（例えば旧バージョン情報など）を付加する。変更の場合には、編集前の要素情報を削除、編集後の要素情報を追加する処理を行えばよい。

【0032】

さらに、編集された部分構造に対応する構造情報の再構築を構造情報登録部 12 において行う。この場合、再構築する構造情報は部分構造の範囲内だけでよく、構造化文書全体の構造情報を再構築する場合に比べて格段に少ない処理で済む。また、他の部分構造に影響しないので、他の部分構造を他人が編集中であっても全く問題なく再構築の処理を行うことができる。

【0033】

また、文書入力部 1 からファイル管理部 2 に対して、文書構造の検索を要求することができる。要求を受け取った検索部 4 では、リレーショナルデータベース 13 に登録されている構造化文書の中から検索条件に適合した文書構造を有する文書を検索して、文書入力部 1 に検索結果を返す。この文書構造の検索の際には、多くの場合、2つの要素間の先祖・子孫関係の検索が行われる。上述のように構造化文書を部分構造に分解して、部分構造間の関係を示すグローバル構造情報と、各部分構造内の要素間の関係を示す構造情報とを有していることによって、階層や枝の多い、大規模な構造化文書でも、容易に検索を行うことができる。

【0034】

まず、同じ部分構造内に存在する要素間の先祖・子孫関係の検索は、当該要素が存在する部分構造の構造情報を参照すればよい。部分構造であるので、要素数も少なく、高速に先祖・子孫関係の検索を行うことができる。なお、部分構造内

での先祖・子孫関係の検索方法は任意であり、従来から用いられている各種の文書構造の検索手法を用いることができる。

【0035】

関係を検索する2つの要素が異なる部分構造に存在している場合には、まず、それぞれの要素が存在する部分構造間の先祖・子孫関係の検索を行う。この検索には、グローバル構造情報を参照すればよい。部分構造の数は要素の数に比べて格段に少なく、この検索は高速に実行可能である。またこの場合の検索手法は、それぞれの部分構造を1つの要素と考えれば従来から用いられている各種の文書構造の検索手法を用いることができる。

【0036】

それぞれの要素が存在する部分構造が先祖・子孫の関係にある場合には、さらに、先祖側の部分構造から子孫側の部分構造へのパス上にあつて、先祖側の部分構造の子の部分構造のルートとなる要素（すなわちマウントポイント）と先祖側の部分構造に含まれる要素との間の先祖・子孫関係を、先祖側の部分構造の構造情報によって求める。

【0037】

例えば図2（A）に示した例において、 $SN_o = 6$ の要素と $SN_o = 9$ の要素との先祖・子孫関係を求める場合、 $SN_o = 6$ の要素が含まれる $GID = 0$ の部分構造と、 $SN_o = 9$ の要素が含まれる $GID = 2$ の部分構造とは先祖・子孫の関係にあり、さらに、 $GID = 2$ の部分構造のルートとなる要素（ $SN_o = 8$ であり $SN_o = 14$ でもある）と $SN_o = 6$ との先祖・子孫関係を、 $GID = 0$ の部分構造の構造情報を用いて調べる。これは先祖・子孫の関係にあるため、結果として、 $SN_o = 6$ と $SN_o = 9$ とは先祖・子孫関係にあると判断される。

【0038】

同じ $GID = 0$ の部分構造の要素でも、 $SN_o = 2$ の要素の場合、部分構造間の関係は先祖・子孫関係となるものの、 $SN_o = 2$ と $SN_o = 14$ との関係は先祖・子孫関係にないため、 $SN_o = 2$ と $SN_o = 9$ の要素は先祖・子孫関係にないと判断される。

【0039】

このように、部分構造間の先祖・子孫関係を調べた後に、先祖側の部分構造内の関係を調べるだけで、2つの要素間の先祖・子孫関係を調べることができる。要素数が膨大な、大規模な文書においては、従来のような要素を1つずつたぐってゆく検索手法では、このような先祖・子孫関係の検索は非常に時間のかかる処理である。しかし本発明によれば、短時間で、簡単に、先祖・子孫関係の検索を行うことができる。

【0040】

なお、先祖側の部分構造に含まれる要素が、その部分構造のルートとなる要素である場合には、部分構造間の先祖・子孫関係の検索を行うだけでよい。なぜなら、その部分構造のルートとなる要素であれば、その部分構造の下位に存在するいずれの部分構造内の要素の親であることが明白だからである。このように後続の判定を行わないことによって、検索処理をさらに高速化することができる。

【0041】

以下、具体例を用いながら上述の動作についてさらに詳細に説明してゆく。図4は、具体例におけるデータ構造の一例の説明図である。以下に説明する具体例では、図4に示すようなデータ構造を用いてグローバル構造情報、要素情報、構造情報をリレーショナルデータベース13に登録し、利用する。

【0042】

まずグローバル構造情報は、図4(A)に示すように、それぞれの部分構造ごとに、最大GID、親GID、親の接続ID、編集単位IDなどの情報を有し、これらの部分構造ごとのデータをGIDの順に並べて構成されている。GIDは、ルートノードから深さ優先、左優先で探索される部分構造に対して順に振られたIDである。このGIDの振り方は、後述する部分構造内の構造情報において各要素に振るノードオーダーと同様であり、後述するノードオーダーの振り方の説明によりGIDの振り方の説明に代える。最大GIDは、当該部分構造よりも下位の部分構造のGIDのうち最大のGIDである。深さ優先でGIDを振っているので、当該部分構造の下位の部分構造には、当該部分構造のGIDから最大GIDまでのGIDが振られている。また、最大GIDよりも大きいGIDは、当該部分構造の下位には存在しないことが保証されている。親GIDは、当該部

分構造の親となる部分構造のG I Dである。親の接続I Dは、例えば図2（B）に示す例ではG I Dが1の部分構造は、G I Dが0の部分構造において仮想的に付加した要素のうちのいずれに接続されるかを示すものである。編集単位I Dは、当該部分構造をシステム全体で一意に識別するためのI Dである。これらの情報は、例えばG I Dの数だけ並べてバイナリ形式でレコード中の1つのフィールド値とし、そのほかに文書I Dなどのフィールドを付加して、リレーショナルデータベース13に保持させることができる。もちろん、保存時のデータ形式は任意である。

【0043】

要素情報は、図4（B）に示すように、ノードテーブル、属性テーブル、テキストテーブルによって構成される。要素のうち、リーフとなる要素についてはテキストテーブルの情報をもち、その他の要素についてはノードテーブル及び属性テーブルの情報を持つことになる。例えばノードテーブルと属性テーブルを一つのテーブルにするなど、変形は任意である。

【0044】

ノードテーブルは、上述のS N o、L N o、編集単位I D、要素名、文書I D、追加バージョン情報、削除バージョン情報、マウントポイント数などのフィールドを有している。S N oはシステム全体において一意に要素に振られたI Dである。L N oは、部分構造内で一意に要素に振られたI Dである。このL N oは、文書構造とは無関係である。編集単位I Dは、当該要素を含んでいる部分構造を特定するI Dである。要素名は、文書中の要素の名前である。文書I Dは、当該要素を含んでいる文書を特定するI Dである。追加バージョン情報は、当該要素が追加されたバージョンを示す情報である。新規に文書が登録された時点では1となる。削除バージョン情報は、当該要素が削除される直前のバージョンを示す情報である。新規に文書が登録された時点ではN U L Lであり、例えば最初の更新で削除されれば1となる。マウントポイント数は、当該要素の下位（子孫）に接続されるマウントポイントの数を示す。マウントポイントは、上述のように複数の部分構造にまたがる要素について、位の部分構造に付加したダミーの要素である。例えば図2（B）に示すG I Dが0の部分構造では、S N oが1の要素

のマウントポイント数は2となる。また属性テーブルは、S N o、属性名、属性値などのフィールドを有している。S N oはノードテーブルと同様である。属性名は当該要素の属性に付された名前であり、その値が属性値として格納される。

【0045】

テキストテーブルは、S N o、L N o、編集単位ID、要素値、文書ID、追加バージョン情報、削除バージョン情報などのフィールドを有している。要素値には、リーフとなる要素が有している文書内容が格納される。また、例えば図2 (B) に示す例においてG I Dが0の部分構造に仮想的に付加した要素は、その部分構造においてはリーフとなる要素である。この場合には、要素値として接続IDが格納される。他のフィールドについては上述の通りである。

【0046】

構造情報は、図4 (C) に示すように、部分構造内の各要素ごとにL N o、ノードオーダー、最大ノードオーダーなどのデータが並べられている。L N oは上述の通りである。ノードオーダーは、部分構造内のルートとなる要素から深さ優先、左優先で探索される要素に対して順に振られたIDである。最大ノードオーダーは、部分構造中で当該要素よりも下位の要素のノードオーダーのうち最大のノードオーダーである。深さ優先でノードオーダーを振っているので、当該要素の下位の要素には、当該要素のノードオーダーから最大ノードオーダーまでのIDが振られている。また、最大ノードオーダーよりも大きいIDは、当該要素の下位には存在しないことが保証されている。これらの情報は、例えば、部分構造内に存在する要素の数（仮想的に設けた要素も含めて）だけ並べてバイナリ形式でレコード中の1つのフィールド値とし、そのほかにクラスタIDなどのフィールドを付加してリレーショナルデータベース13に保持させることができる。もちろん、保存時のデータ形式は任意である。なお、クラスタIDは、上述の編集単位IDと同様に部分構造を特定するIDであるが、構造情報は文書の編集などにより版が変わったときに再構築されるので、版が変わるごとに新たなクラスタIDが付与される。バージョン（版）管理を行わない場合には、クラスタIDは編集単位IDでよい。

【0047】

バージョンテーブルは、図4（D）に示すように、クラスタID、編集単位ID、バージョン情報、文書IDなどによって構成されている。このバージョンテーブルによって、グローバル構造情報において部分構造に付した編集単位IDと、バージョンごとに異なる部分構造内の文書構造を示す構造情報とを結びつけている。なお、別途、各文書について最新のバージョンを示す情報が保持されており、最新の文書を取り出す際にはその最新バージョン情報からバージョンテーブルを検索することによって、各部分構造についての最新の構造情報を得ることができる。また、属性情報については、上述のように削除バージョン情報がNULLのレコードを取り出せば、最新の要素情報を取り出すことができる。バージョン（版）管理を行わない場合には、このバージョンテーブルを設けずに構成することもできる。

【0048】

図5は、入力される構造化文書の具体例の説明図である。ここでは、XMLによって記述された、図5（A）に示すような構造化文書が文書入力部1から入力されるものとする。この構造化文書の文書構造をツリー表現すると図5（B）に示すようになる。なお、図5（B）において○は要素を示し、内部の数字はSN_oを示している。抜けているSN_oについては、後述するようにマウントポイントに割り当てられる。

【0049】

このような構造化文書が文書入力部1から入力されると、ディスプレイ3は、入力された構造化文書を文書処理部5へ渡す。文書処理部5では、文書解析部6で構造化文書を解析し、文書分解部7でリレーショナルデータベース13から読み込んだ設定に従い、文書を複数の部分構造に分解する。ここではCHAPTER、SECTIONで分解することとする。

【0050】

図6～図8は、分解された部分構造ごとの文書の具体例の説明図である。各図において、分解された文書の記述を（A）として示し、分解された部分構造のツリー表現を（B）として示している。以下の説明では図6に示す部分構造を編集単位ID=0、図7に示す部分構造を編集単位ID=1、図8に示す部分構造を

編集単位ID=2とする。また、この編集単位IDとは別に、各部分構造にはGIDが振られる。ここでは、図6に示す部分構造にGID=0、図7に示す部分構造にGID=1、図8に示す部分構造にGID=2を振るものとする。

【0051】

図6に示す部分構造では、部分構造への分解の際に、図5(B)に示すSNo=6の要素に対応する仮想的な要素(SNo=16)を付加している。図6(A)に示す文書の記述では、分解により他の部分構造に移した記述の部分を“&E001;”という記述に置き換えている。これが接続IDとなる。図7に示す部分構造についても同様であり、図5(B)に示すSNo=9の要素に対応する仮想的な要素(SNo=18)を付加している。図7(A)に示す文書の記述では、分解により他の部分構造に移した記述の部分を“&E002;”という接続IDに置き換えている。この置き換えられた要素がマウントポイントとなる。

【0052】

図9は、グローバル構造情報の具体例の説明図である。上述のように構造化文書を部分構造に分解すると、その時点で各部分構造間の関係がわかる。各部分構造間のツリー表現を図9(B)に示している。図9(B)において、各部分構造を三角形で表している。三角形の中には、GIDとともにマウントポイントの接続IDを示している。

【0053】

この部分構造間の関係をグローバル構造情報としてリレーショナルデータベース13に登録する。この具体例では、グローバル構造情報は例えば図9(A)に示すようになる。なお、GID=1の部分構造は、GID=0の部分構造中のマウントポイント(SNo=16)に接続されるので、その接続ID(“&E001;”)のうちの数値部分のみを親の接続IDとしてグローバル構造情報中に含めている。GID=2の部分構造についても同様であり、親の接続IDとしてGID=1の部分構造中のマウントポイント(SNo=18)の接続ID(“&E002;”)のうちの数値部分のみを親の接続IDとしてグローバル構造情報中に含めている。

【0054】

このようなグローバル構造情報は、文書IDを付加してリレーショナルデータベース13に登録する。このとき、例えばバイナリ形式のままフィールド値として格納しておくことによって、検索時のパフォーマンスを向上させることができる。

【0055】

図6(A)、図7(A)、図8(A)に示した、分解された部分構造の記述は、再びファイル管理部2に戻される。ファイル管理部2のディスパッチャ3は、これらの分解された部分構造の記述を編集単位処理部8に渡す。編集単位処理部8では、文書解析部9で各部分構造の記述を解析し、差分解部10に渡す。差分解部10では、新規の構造化文書であるので、文書解析部9による部分構造の解析結果を要素情報登録部11及び構造情報登録部12に渡す。

【0056】

要素情報登録部11では、各部分構造の記述から、要素情報を生成する。図4(B)に示した要素情報のデータ構造に従い、リーフに対応する要素についてはテキストテーブルを生成し、それ以外の要素についてはノードテーブル及び属性テーブルを生成する。図10、図11は、要素情報の具体例の説明図である。図10(A)はノードテーブル、図10(B)は属性テーブル、図11はテキストテーブルを示している。SNoは、図6～図8では説明のためにすでに振られているが、実際には編集単位処理部8に各部分構造の記述が入力されて解析することによって、要素の出現順に振られる。LNNoについても、各部分構造内で文書構造には関係なく振られる。追加バージョンには、新規登録時には1が格納される。また削除バージョンには、新規登録時には‘NULL’が格納される。マウントポイント数には、仮想的な要素(SNo=12, 14)が下位に接続されているSNo=0, 3, 6の要素について‘1’となり、他の要素については0となる。このような要素情報がリレーショナルデータベース13に登録される。

【0057】

構造情報登録部12では、部分構造中の各要素間の関係を示す構造情報を生成する。図4(C)に示した構造情報のデータ構造に従い、それぞれの要素のLNNo、ノードオーダー、最大ノードオーダーを求めて格納してゆく。図12は、構

造情報を生成する処理の一例を示すフローチャートである。ここではXMLのタグ及びテキストが順次切り出され、そのたびにそれぞれの処理を行うものとし、またLN○についても別途付与されるものとして説明する。なお、処理を進めるためにLN○及びインデックスの値を格納するためのスタックを用いる。

【0058】

S21において、初期設定としてインデックスを0にセットしておく。S22において、タグまたはテキストの切出を検知し、LN○を取得する。S23において、タグまたはテキストが存在していたか否かを判定し、タグまたはテキストを切り出すことができず、部分構造の記述が終了していればこの処理を終了する。

【0059】

S24において、切り出された要素がテキストか否かを判断し、切り出された要素がタグであればさらにS25において開始タグか終了タグかを判定する。切り出された要素が開始タグの場合には、S26において、S22で取得したLN○と現在のインデックスをスタックに積む。そしてS27において、インデックスの値を1だけ増加させる。

【0060】

切り出された要素がテキストの場合には、S28において、現在のインデックスの値をノードオーダー及び最大ノードオーダーとし、S22で取得したLN○に対応する構造情報の位置に、LN○、ノードオーダー、最大ノードオーダーを格納する。

【0061】

切り出された要素が終了タグの場合には、S29において、スタックからLN○及びインデックスを取り出し、S30において、ノードオーダーとしてスタックから取り出したインデックスの値、最大ノードオーダーとして現在のインデックスの値をセットし、スタックから取り出したLN○に対応する構造情報の位置に、LN○、ノードオーダー、最大ノードオーダーを格納する。

【0062】

このような処理を行うと、ノードオーダーとして、それぞれの要素には深さ優

先、左優先で探索された順に番号付けが行われる。図 1 3 は、編集単位 ID が 2 の部分構造における構造情報生成過程の一例の説明図である。図 1 0 (A) 及び図 1 1 に示したように、各要素には L N o が振られている。この L N o は文書構造とは関係なく振られている。図 8 (A) に示すような編集単位 ID が 2 の XML による記述をもとに、図 1 2 に示した処理を実行する。なお、図 1 3 において矩形で囲んだ 3 つ組の数値は、左から L N o、ノードオーダー、最大ノードオーダーである。

【 0 0 6 3 】

L N o = 0, 1, 2 については L N o とそのときのインデックス (0, 1, 2) がスタックに積まれる。L N o = 5 のリーフノードの要素において、L N o = 5、ノードオーダー = 3、最大ノードオーダー = 3 が構造情報として格納される。スタックから L N o = 2 とインデックス = 2 が取り出され、L N o = 2、ノードオーダー = 2、最大ノードオーダー = 3 が構造情報として格納される。さらにスタックから L N o = 1 とインデックス = 1 が取り出され、L N o = 1、ノードオーダー = 1、最大ノードオーダー = 3 が構造情報として格納される。

【 0 0 6 4 】

次に L N o = 3, 4 についてはスタックに当該 L N o とインデックス = 4, 5 が積まれる。L N o = 6 のリーフノードの要素において、L N o = 6、ノードオーダー = 6、最大ノードオーダー = 6 が構造情報として格納される。スタックから L N o = 4 とインデックス = 5 が取り出され、L N o = 4、ノードオーダー = 5、最大ノードオーダー = 6 が構造情報として格納される。さらにスタックから L N o = 3 とインデックス = 4 が取り出され、L N o = 3、ノードオーダー = 4、最大ノードオーダー = 6 が構造情報として格納される。さらにまた、スタックから L N o = 0 とインデックス = 0 が取り出され、L N o = 0、ノードオーダー = 0、最大ノードオーダー = 6 が構造情報として格納される。

【 0 0 6 5 】

図 1 4 は、構造情報及びバージョンテーブルの具体例の説明図である。上述のような処理によって、各部分構造ごとに図 1 4 (A) ~ (C) に示すような構造情報が得られる。特に編集単位 ID が 2 の部分構造においては、図 1 3 を用いて

詳述したような処理によって、図14(C)に示すような構造情報が得られることになる。図14では構造情報をテーブル形式で表現しているが、これらのデータはバイナリ形式でレコード中のフィールドに格納され、さらにクラスタIDが付されてリレーショナルデータベース13に登録される。また、図14(D)に示すように、バージョンごとにクラスタIDと編集単位IDとを対応づけるバージョン情報のレコードが各部分構造ごとに生成され、リレーショナルデータベース13に登録される。

【0066】

以上説明したようにして、図5に示した構造化文書は、図6～図8に示したように部分構造に分割され、部分構造の間の関係を示すグローバル構造情報と、各要素の情報を格納した要素情報と、部分構造内の各要素間の関係を示す構造情報、それに編集時に備えてバージョンテーブルが生成され、リレーショナルデータベース13に登録される。

【0067】

なお、文書の編集は部分構造ごとに行われる。編集後の部分構造はファイル管理部2に入力され、ディスパッチャ3が編集単位処理部8へ渡す。編集単位処理部8では、渡された編集後の部分構造について、文書解析部9で解析し、差分解析部10は解析結果をもとにリレーショナルデータベース13に登録されている編集前の部分構造との差分を抽出する。そして、追加あるいは編集されて変更された後の要素については新たなバージョン情報を追加バージョン情報とした要素情報を生成し、リレーショナルデータベース13に登録する。また削除あるいは編集されて変更される前の要素については、編集直前のバージョン情報を削除バージョン情報として当該フィールドに書き込んでおく。また、編集された部分構造の構造情報を再構築し、新たなクラスタIDを付してリレーショナルデータベース13に登録し、また、対応するバージョンテーブルのレコードを生成して同じくリレーショナルデータベース13に登録する。

【0068】

このようにして編集が加えられた場合のバージョン管理を行うことができる。このようなバージョン管理によって、例えば最新版の文書の要素情報は削除バー

ジョンがNULLのレコードを取り出せばよく、簡単に最新版の文書あるいは部分構造を取り出すことができる。もちろん、このようなバージョン管理を行わないで構成することも可能である。その場合には、データ構造をより簡略化することができる。

【0069】

次に、文書構造の検索を行う場合について具体例を用いて説明する。文書構造の検索を行う場合、その検索条件としては、ある属性を有する要素の下位に、ある属性を有する要素が存在する文書構造、といった条件が設定される場合が考えられる。ここではそのような検索条件による文書構造の検索を考え、一例として、要素の属性条件 a を満たす要素の集合 $A \{a_i \mid 1 \leq i \leq N_a\}$ (ただし N_a は属性条件 a を満たす要素数。以下 $\{a_i\}$ と記す。) に含まれる1つの要素が、要素の内容条件 b を満たす要素の集合 $B \{b_j \mid 1 \leq j \leq N_b\}$ (ただし N_b は属性条件 b を満たす要素数。以下 $\{b_j\}$ と記す。) のうち、いずれか一つ以上の要素を子孫に持つ集合を求める場合の動作について説明する。なお、要素の属性条件 a を満たす要素の集合 $A \{a_i\}$ 、及び、要素の内容条件 b を満たす要素の集合 $B \{b_j\}$ は、例えば図10(A)に示すノードテーブル及び図11に示すテキストテーブルなどの要素情報に対して、属性値のフィールドで検索を行うことによって容易に求めることができる。

【0070】

図15は、本発明の実施の一形態における2要素間の先祖・子孫関係の検索処理の一例を示すフローチャートである。ここでは集合 $A \{a_i\}$ に含まれる1つの要素 a_i と、集合 $B \{b_j\}$ に含まれる1つの要素 b_j との間の先祖・子孫関係を判定する処理を示している。なお、要素 a_i が含まれる部分構造を C_{a_i} 、要素 b_j が含まれる部分構造を C_{b_j} とする。また、要素 a_i については、その要素 a_i が含まれる部分構造 C_{a_i} において、要素 a_i がルートとなる要素であるか否かが、わかっているものとする。

【0071】

まずS41において、要素 a_i と要素 b_j が同じ文書に含まれているか否かを判定する。この判定は、要素情報中の文書IDを比較すればよい。別の文書であ

れば以降の処理を行わずに、先祖・子孫関係にはないと判定する。なお、例えば要素 b_j を文書 ID ごとに昇順にソートしておく、要素 a_i の文書 ID を上回った時点で以後の要素 b_j については先祖・子孫関係にはないと判定して処理を終了することができる。もちろん、降順にソートしておいて、要素 a_i の文書 ID を下回った時点で処理を終了しても同様である。

【0072】

要素 a_i と要素 b_j が同じ文書に含まれている場合、まず S42において、要素 a_i が含まれる部分構造 C_{a_i} と、要素 b_j が含まれる部分構造 C_{b_j} が同一の部分構造か否かを判定する。この判定は、部分構造 C_{a_i} と部分構造 C_{b_j} のクラスタ ID が一致するか否かを判定することにより行うことができる。部分構造 C_{a_i} と部分構造 C_{b_j} が同一である場合には、要素 a_i と要素 b_j は同じ部分構造内の要素である。この場合、当該部分構造内で要素 a_i と要素 b_j の先祖・子孫関係を判定すればよい。

【0073】

このとき、S43において、要素 a_i が部分構造 C_{a_i} のルートノードであるか否かを判定する。要素 a_i がルートノードである場合には、同じ部分構造内の要素 b_j は必ず要素 a_i の子孫である。従って、要素 a_i と要素 b_j は先祖・子孫関係にあると判定する。

【0074】

要素 a_i が部分構造 C_{a_i} のルートノードではない場合には、S44において、要素 a_i と要素 b_j の先祖・子孫関係を判定する。この場合には要素 a_i と要素 b_j が同じ部分構造内であるので、当該部分構造の構造情報を用いて先祖・子孫関係を判定することができる。判定は、要素 a_i のノードオーダー及び最大ノードオーダーと、要素 b_j のノードオーダーを用い、

$(\text{要素 } a_i \text{ のノードオーダー}) < (\text{要素 } b_j \text{ のノードオーダー}) \leq (\text{要素 } a_i \text{ の最大ノードオーダー})$

が成り立てば、要素 b_j は要素 a_i の子孫であると判定できる。この条件が満たされない場合、要素 b_j は要素 a_i の子孫ではないと判定できる。

【0075】

要素 a_i が含まれる部分構造 C_{a_i} と、要素 b_j が含まれる部分構造 C_{b_j} が異なる部分構造である場合には、基本的には部分構造 C_{a_i} と部分構造 C_{b_j} の先祖・子孫関係を調べ、その後、要素 a_i と部分構造 C_{a_i} 内のマウントポイントとの関係を調べる。まず S45 において、要素 a_i のマウントポイント数を調べる。マウントポイント数が 0 であれば、下位の部分構造には要素 a_i の子孫が存在しないことが明らかであるので、この時点で要素 a_i と要素 b_j は先祖・子孫関係にはないと判定できる。この判定によって、処理の高速化を図っている。またこの場合には、部分構造 C_{a_i} に含まれない要素 b_j については、すべて先祖・子孫関係にはない。従って、要素 a_i に対する検索においては、集合 B のうち、部分構造 C_{a_i} に含まれない要素 b_j については検索不要としてフラグを立てておくといよい。

【0076】

要素 a_i のマウントポイント数が 0 でないときには、S46 において、部分構造 C_{a_i} と部分構造 C_{b_j} との先祖・子孫関係を調べる。グローバル構造情報を参照して部分構造 C_{a_i} の GID、最大 GID と、部分構造 C_{b_j} の GID を取得し、

$$(C_{a_i} \text{ の GID }) < (C_{b_j} \text{ の GID }) \leq (C_{a_i} \text{ の最大 GID })$$

を判定する。この条件が成り立つとき、部分構造 C_{a_i} と部分構造 C_{b_j} とは先祖・子孫関係にあると言える。この条件が成り立たないとき、部分構造 C_{a_i} と部分構造 C_{b_j} とは先祖・子孫関係ではなく、従って要素 a_i と要素 b_j も先祖・子孫関係ではないと判定する。なお、部分構造 C_{a_i} と部分構造 C_{b_j} が先祖・子孫関係でない場合には、部分構造 C_{a_i} に含まれるすべての集合 A の要素について、部分構造 C_{b_j} に含まれる集合 B の検索は不要であるというフラグを立てておくといよい。これによって不要な検索処理を行わずに済み、処理の高速化を図ることができる。

【0077】

部分構造 C_{a_i} と部分構造 C_{b_j} とが先祖・子孫関係にある場合には、次に要素 a_i と部分構造 C_{a_i} 内のマウントポイントとの関係を調べるが、処理量を軽減するために、S47 において、要素 a_i が部分構造 C_{a_i} のルートノードであ

るか否かを判定する。要素 a_i がルートノードである場合には、要素 a_i は、部分構造 C_{a_i} の下位の部分構造に含まれるすべての要素の先祖であることが必ず言える。従って、部分構造 C_{a_i} と部分構造 C_{b_j} とが先祖・子孫関係であり、かつ、要素 a_i が部分構造 C_{a_i} のルートノードである場合には、要素 a_i と要素 b_j は先祖・子孫関係にあると判定する。

【0078】

S48では、要素 a_i と部分構造 C_{a_i} 内のマウントポイントとの関係を調べる。部分構造 C_{b_j} から部分構造 C_{a_i} までのパスをたどり、そのパス上にあって、部分構造 C_{a_i} の直下の部分構造のルートノードに対応する部分構造 C_{a_i} におけるマウントポイントを求め、そのマウントポイントのノードオーダー (m とする) を取得する。この処理は、グローバル構造情報を用いて行うことができ、部分構造 C_{b_j} から親 GID をたどってゆき、親 GID が部分構造 C_{a_i} の GID となったとき、その部分構造が部分構造 C_{a_i} の直下の部分構造である。そして、その部分構造 C_{a_i} の直下の部分構造のグローバル構造情報中の親の接続 ID の値を取得することによって、部分構造 C_{a_i} 中のマウントポイントを求めることができる。さらにマウントポイントの要素情報及び構造情報からノードオーダーを取得することができる。

【0079】

そして、要素 a_i のノードオーダーと最大ノードオーダー、それにマウントポイントのノードオーダー m の関係が

$$(\text{要素 } a_i \text{ のノードオーダー}) \leq m \leq (\text{要素 } a_i \text{ の最大ノードオーダー})$$

を満たすか否かを判定する。この条件を満たすとき、要素 a_i とマウントポイントとは先祖・子孫関係にあり、よって、要素 a_i と要素 b_j とは先祖・子孫関係にあると判断することができる。また、この条件が満たされないとき、要素 a_i とマウントポイントとは先祖・子孫の関係にはなく、よって、要素 a_i と要素 b_j とは先祖・子孫関係にはないと判断することができる。さらに、この条件が満たされないときには、要素 a_i の検索において、部分構造 C_{b_j} に含まれる集合 B の要素は検索しなくてよいものとしてフラグを立てておくことによって、検索処理を高速化することができる。

【0080】

上述の処理はある要素 a_i と要素 b_j が特定された場合の先祖・子孫関係の判定処理であるが、このような処理を、集合 A 中のある要素 a_i について、集合 B 中のそれぞれの要素 b_j について行うことになる。要素 b_j の選択の際には、S45, S46, S48 において検索不要のフラグを立てた要素については検索対象から外すことによって、検索処理を高速化できる。

【0081】

さらに、集合 A 中の要素を変更し、同様にして集合 B 内のそれぞれの要素との判定を行ってゆけばよい。このときにも、S46 におけるフラグを考慮することによって、検索処理を高速化することができる。

【0082】

検索結果としては、例えば検索条件が、ある属性を有する要素の下位に、ある属性を有する要素が存在する文書構造の存在するか否かのみである場合には、存在する、あるいは存在しないのみを結果とすればよい。この場合には、ある2つの要素間で先祖・子孫関係が確認できた時点で処理を終了することができる。また、ある属性を有する要素の下位に、ある属性を有する要素が存在する文書構造の実体の検索であれば、例えば条件を満たす集合 A の要素、あるいは要素を含む部分構造、または要素を含む文書 ID、などを検索結果とすればよい。

【0083】

図5以降で説明してきた具体例を用いて、上述の検索処理についていくつかの例を説明する。図16、図17は、2要素間の先祖・子孫関係の検索処理の一例における具体的な動作の一例の説明図である。各図においては、説明に関係する要素部分のみを示している。まず、属性 SECTION の要素の下位に本文の要素を有する文書構造を検索する場合を図16(A)に示している。この場合、属性 SECTION を有する要素は、 $SN_o = 9$ の要素である。また、本文は $SN_o = 20$ の要素である。図10(A)及び図11から、両者とも編集単位 ID が2であり、同じ部分構造（編集単位 ID = 2, GID = 2）に属することがわかる。さらに、 $SN_o = 9$ の要素はこの部分構造のルートノードである。従って、S43において、この2つの要素は先祖・子孫関係にあると判定される。

【0084】

次に、属性DOCUMENTの要素の下位に本文の要素を有する文書構造を検索する場合を図16(B)に示している。この場合、属性DOCUMENTを有する要素は、SNo=12の要素である。また、本文はSNo=20の要素である。図10(A)及び図11から、両者とも編集単位IDが2であり、同じ部分構造(編集単位ID=2, GID=2)に属する。しかし、SNo=12の要素はこの部分構造のルートノードではない。そのため、さらにSNo=12の要素とSNo=20の要素の先祖・子孫関係をS44で判定する。すなわち、図10に示す要素情報からSNo=12のLNoが3であり、図14(C)に示す編集単位ID=2の構造情報からノードオーダーが4、最大ノードオーダーが6であることが求まる。また同様に、図11に示す要素情報(テキストテーブル)からSNo=20のLNoが6であり、図14(C)に示す編集単位ID=2の構造情報からノードオーダーが6であることが求まる。このSNo=20の要素のノードオーダー=6は、SNo=12の要素のノードオーダー=4と最大ノードオーダー=6の範囲内であるので、SNo=12の要素とSNo=20の要素とは先祖・子孫関係にあると判定される。

【0085】

図16(C)に示す例では、属性CHAPTERの要素の下位に本文の要素を有する文書構造を検索する場合を示している。この場合、図10, 図11に示す要素情報の検索から、属性CHAPTERを有する要素は、GID=1の部分構造(編集単位ID=1)に属するSNo=6の要素である。また、本文はGID=2の部分構造(編集単位ID=2)に属するSNo=20の要素である。両者は異なる部分構造に属する。また、図10(A)に示すノードテーブルから、SNo=6の要素は、マウントポイント数が1であり、子孫にマウントポイントが存在する。そして、図9に示すグローバル構造情報から、GID=1の部分構造とGID=2の部分構造は、先祖・子孫関係にある。これらの判定によってS47へ進み、SNo=6の要素がGID=1の部分構造のルートノードであることから、GID=1の部分構造に属するSNo=6の要素と、GID=2の部分構造に属するSNo=20の要素とは先祖・子孫関係にあると判定される。

【0086】

図16 (D) に示す例では、属性PARTの要素の下位に属性DOCITEMの要素を有する文書構造を検索する場合を示している。この場合、図10、図11に示す要素情報の検索から、属性PARTを有する要素は、GID=0の部分構造（編集単位ID=0）に属するSNo=3の要素である。また、属性DOCITEMを有する要素はGID=2の部分構造（編集単位ID=2）に属するSNo=12の要素である。また、要素情報より、両者の編集単位IDは異なり、両者は異なる部分構造に属する。さらに要素情報より、SNo=3の要素のマウントポイント数は1であり、子孫にマウントポイントが存在することがわかる。そして、GID=0の部分構造とGID=2の部分構造は、グローバル構造情報を参照することによって、先祖・子孫関係にあることがわかる。しかし、SNo=3の要素は、GID=0の部分構造におけるルートノードではない。従ってS48の判定を行うことになる。

【0087】

この場合、GID=2のグローバル構造情報の親GIDから、GID=2の部分構造の親はGID=1の部分構造である。同様に、GID=1のグローバル構造情報の親GIDから、GID=1の部分構造の親はGID=0の部分構造である。このパス上でGID=0の部分構造の直下の部分構造、すなわちGID=1の部分構造のルートノードに対応するGID=0の部分構造のマウントポイントを求める。すなわち、GID=1のグローバル構造情報における親の接続IDを取得すると、“001”となっており、属性値としてE001を有する要素を図11に示す要素情報から検索する。すると、SNo=16の要素がマウントポイントであることがわかる。

【0088】

さらに、図10 (A) に示す要素情報からSNo=3の要素のLNが3であり、図14 (A) に示すGID=0の構造情報中のLN=3を参照することによって、SNo=3の要素のノードオーダーが4、最大ノードオーダーが8であることがわかる。同様に図11に示す要素情報からSNo=16の要素のLNが8であり、図14 (A) に示すGID=0の構造情報中のLN=8を参照す

ることにより、 $SN_o = 16$ のマウントポイントのノードオーダーが8であることがわかる。この $SN_o = 16$ のマウントポイントのノードオーダー=8は、 $SN_o = 3$ の要素のノードオーダー=4と最大ノードオーダー=8の範囲内である。従って、 $SN_o = 16$ のマウントポイントと $SN_o = 3$ の要素とは先祖・子孫関係にある。従って、 $GID = 0$ の部分構造に属する $SN_o = 3$ の要素と、 $GID = 2$ の部分構造に属する $SN_o = 12$ の要素とは先祖・子孫関係にあると判定される。

【0089】

図17(A)に示す例では、属性TITLEの要素の下位に属性DOCITEMの要素を有する文書構造を検索する場合を示している。この場合、図10、図11に示す要素情報の検索から、属性TITLEを有する要素は、 $GID = 0$ の部分構造（編集単位ID=0）に属する $SN_o = 1$ の要素と $SN_o = 4$ の要素、 $GID = 1$ の部分構造（編集単位ID=1）に属する $SN_o = 7$ の要素、 $GID = 2$ の部分構造（編集単位ID=2）に属する $SN_o = 10$ の要素の4つである。また、属性DOCITEMを有する要素は $GID = 2$ の部分構造（編集単位ID=2）に属する $SN_o = 12$ の要素である。

【0090】

このうち、 $SN_o = 1, 4, 7$ の要素については、 $SN_o = 12$ の要素とは異なる部分構造に属している。しかし、 $SN_o = 1, 4, 7$ の要素は、図10(A)に示す要素情報からマウントポイント数が0であり、S45において $SN_o = 12$ の要素との先祖・子孫関係は否定される。

【0091】

また、 $SN_o = 10$ の要素については、 $SN_o = 12$ の要素と同じ部分構造（編集単位ID=2、 $GID = 2$ ）に属している。さらに $SN_o = 10$ の要素は、この部分構造のルートノードではない。そのため、S44において $SN_o = 10$ の要素と $SN_o = 12$ の要素の先祖・子孫関係の判定を行う。 $SN_o = 10$ の要素に対応するノードオーダーと最大ノードオーダーは、図14(C)に示す編集単位ID=2の構造情報から、それぞれ1, 3である。また同様に $SN_o = 12$ の要素に対するノードオーダーは4である。この $SN_o = 12$ の要素に対するノ

ードオーダー=4は、 $SN_o=10$ の要素に対応するノードオーダー=1と最大ノードオーダー=3の範囲に入っていないので、 $SN_o=10$ の要素と $SN_o=12$ の要素との先祖・子孫関係は否定される。

【0092】

図5以降で説明した具体例では、図15のS46、S48で先祖・子孫関係が否定されることはないが、例えば図2に示したような文書構造の場合にあり得る。図17(B)には、属性「章題」の要素の下位に属性「図表」の要素を有する文書構造を検索する場合を示している。以下の説明では詳細は省略するが、属性「章題」の要素として $SN_o=2$ の要素が検索され、属性「図表」の要素として $SN_o=12$ の要素が検索される。この2つの要素は、それぞれ $GID=0$ の部分構造と $GID=2$ の部分構造に含まれており、異なる部分構造に含まれていることがわかる。また、 $SN_o=2$ の要素の下位には、マウントポイントが存在している。さらに、 $GID=0$ の部分構造と $GID=2$ の部分構造とは、先祖・子孫関係にある。従って、図15のS48の判定を行うが、 $GID=2$ の部分構造のルートノードに対応するマウントポイントは $SN_o=14$ の要素であり、 $SN_o=2$ とは先祖・子孫関係にはない。従って、 $SN_o=2$ の要素と $SN_o=12$ の要素との先祖・子孫関係は否定される。

【0093】

図17(C)に示した例では、図2に示した文書構造において、属性「タイトル」の要素の下位に属性「図表」の要素を有する文書構造を検索する場合を示している。やはり詳細な説明は省略するが、属性「タイトル」の要素として $SN_o=3$ の要素が検索され、属性「図表」の要素として $SN_o=12$ の要素が検索される。この2つの要素は、それぞれ $GID=1$ の部分構造と $GID=2$ の部分構造に含まれており、異なる部分構造に含まれていることがわかる。この例では $SN_o=3$ の要素の下位にはマウントポイントが存在していないので、S45の判定によって $SN_o=3$ の要素と $SN_o=12$ の要素との先祖・子孫関係は否定される。仮に $SN_o=3$ の要素の下位にマウントポイントが存在していた場合でも、 $GID=1$ の部分構造と $GID=2$ の部分構造とが先祖・子孫関係にはないので、S46の判定によって $SN_o=3$ の要素と $SN_o=12$ の要素との先祖・子

孫関係は否定される。

【0094】

以上のように、構造化文書を部分構造に分解して、部分構造間の関係を示すグローバル構造情報と、部分構造内の要素間の関係を示す構造情報を保持しているので、上述のように文書構造の検索を行うに際して、2つの要素間の先祖・子孫関係を判定する処理を、高速に実行することができる。

【0095】

なお、上述の検索処理の説明では、文書のバージョンについては触れていないが、例えば検索条件としてバージョンに関する条件を付加することによって、任意のバージョンに対する文書構造の検索を行うことが可能である。また、上述の検索処理の具体例では、図4に示したデータ構造に従った検索処理の一例を示しているが、図4に示したデータ構造は一具体例であって、他のデータ構造を適用可能であり、その場合の検索処理についても本発明の趣旨を逸脱しない範囲で任意の検索手法を適用可能である。

【0096】

上述の実施の形態では、構造に関する情報を部分構造間の関係を示すグローバル構造情報と要素間の関係を示す構造情報で管理構成する構成を示したが、さらに、本発明を多段階に分割したグローバル構造情報に適用することで、さらに大規模な構造化文書の検索を高速に実施できるような構成も可能である。

【0097】

【発明の効果】

以上の説明から明らかなように、本発明によれば、構造化文書を部分構造に分解して、部分構造間の関係を示す第1の構造情報（グローバル構造情報）と、部分構造内の要素間の関係を示す第2の構造情報（構造情報）を保持するようにしたので、例えば文書の編集を行う場合でも、部分構造単位での編集を行うことによって、編集の影響を部分構造内にとどめることができる。そのため、例えば部分構造ごとに編集を行えば並行して複数人が編集を行うことができ、共同編集環境を提供することができる。また、編集による構造情報の再構築を部分構造内にとどめることができ、構造情報の再構築に要する時間を短縮することができる。

【0098】

また、構造化文書に対して頻繁に行われる文書構造の検索に対しても、2要素間の先祖・子孫関係の判定を、部分構造間の先祖・子孫関係の判定と、先祖側の部分構造内の要素とマウントポイントとの判定を行うだけで実行できる。従って、従来のように1つずつ要素をたぐってゆくなどの処理が簡略化され、高速に2要素間の先祖・子孫関係の判定を行って、文書構造の検索の高速化を図ることができるという効果がある。

【図面の簡単な説明】

- 【図1】 本発明の実施の一形態を示すブロック図である。
- 【図2】 構造化文書の構造の一例の説明図である。
- 【図3】 グローバル構造情報及び要素情報と構造情報の一例の説明図である。
- 【図4】 具体例におけるデータ構造の一例の説明図である。
- 【図5】 入力される構造化文書の具体例の説明図である。
- 【図6】 分解された部分構造ごとの文書の具体例（編集単位ID＝0）の説明図である。
- 【図7】 分解された部分構造ごとの文書の具体例（編集単位ID＝1）の説明図である。
- 【図8】 分解された部分構造ごとの文書の具体例（編集単位ID＝2）の説明図である。
- 【図9】 グローバル構造情報の具体例の説明図である。
- 【図10】 要素情報（ノードテーブル及び属性テーブル）の具体例の説明図である。
- 【図11】 要素情報（テキストテーブル）の具体例の説明図である。
- 【図12】 構造情報を生成する処理の一例を示すフローチャートである。
- 【図13】 編集単位IDが2の部分構造における構造情報生成過程の一例の説明図である。
- 【図14】 構造情報及びバージョンテーブルの具体例の説明図である。
- 【図15】 本発明の実施の一形態における2要素間の先祖・子孫関係の検

索処理の一例を示すフローチャートである。

【図 1 6】 2 要素間の先祖・子孫関係の検索処理の一例における具体的な動作（先祖・子孫関係にある場合）の一例の説明図である。

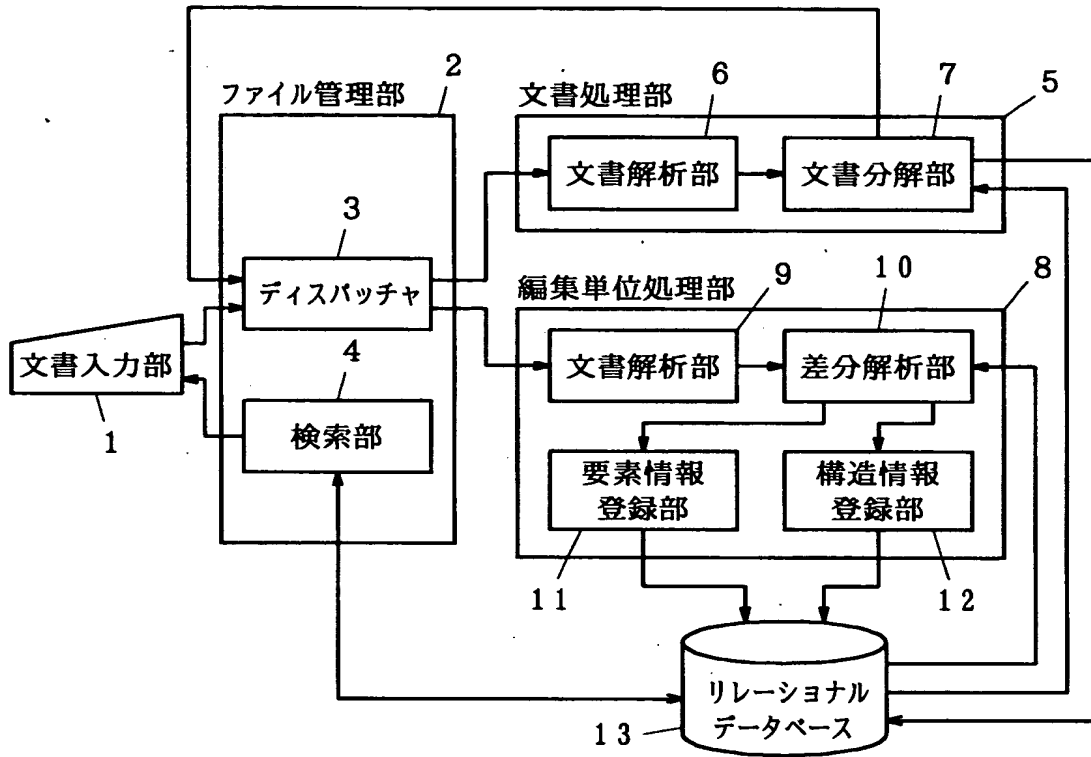
【図 1 7】 2 要素間の先祖・子孫関係の検索処理の一例における具体的な動作（先祖・子孫関係にない場合）の一例の説明図である。

【符号の説明】

1 … 文書入力部、2 … ファイル管理部、3 … ディスパッチャ、4 … 検索部、5 … 文書処理部、6 … 文書解析部、7 … 文書分解部、8 … 編集単位処理部、9 … 文書解析部、10 … 差分解部、11 … 要素情報登録部、12 … 構造情報登録部、13 … リレーショナルデータベース。

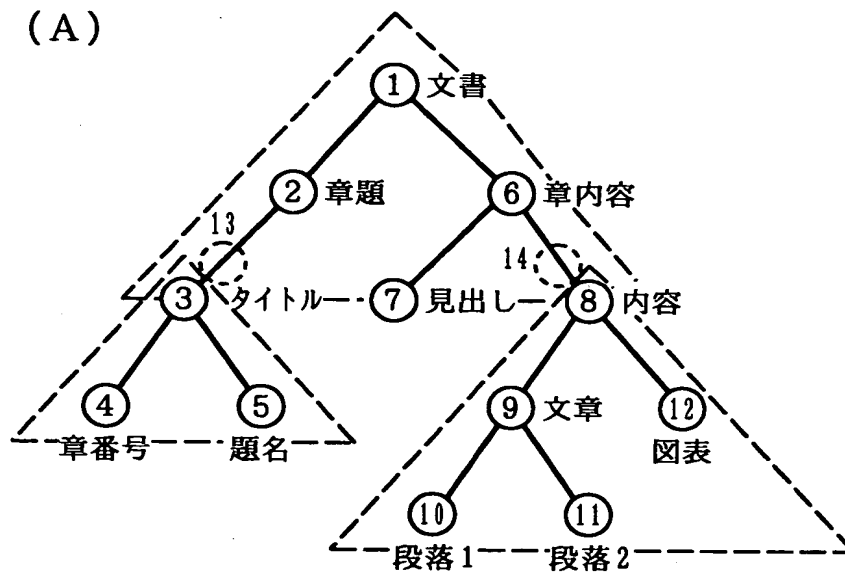
【書類名】 図面

【図1】

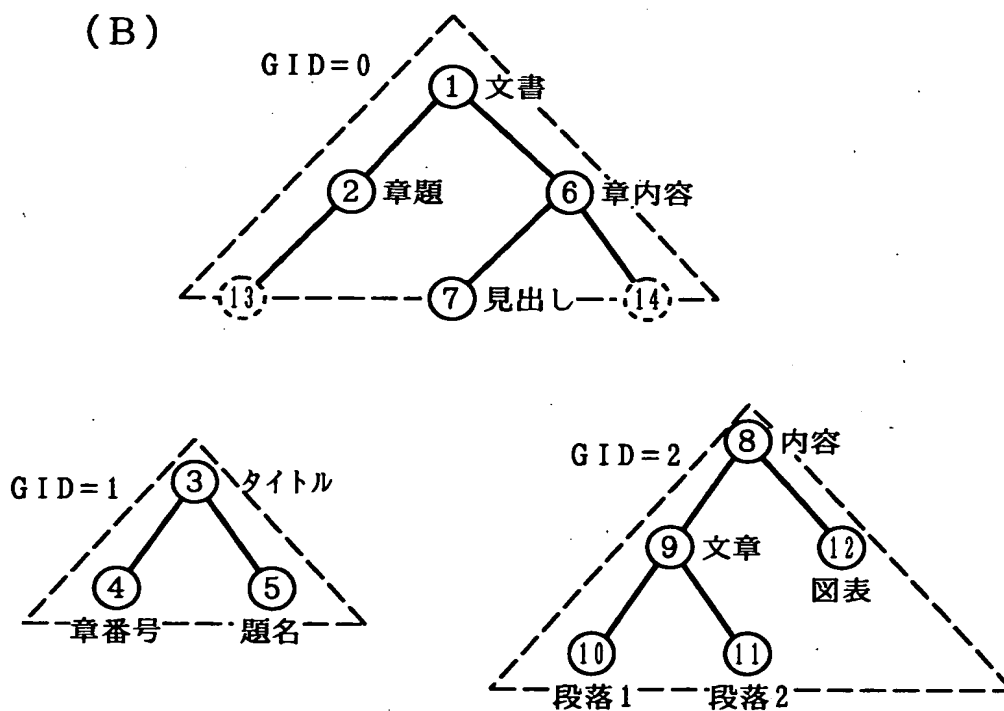


【図 2】

(A)

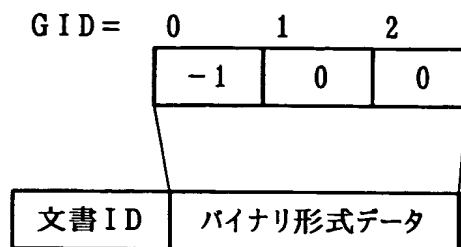


(B)



【図3】

(A)

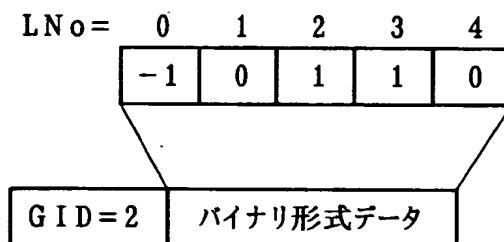


(B)

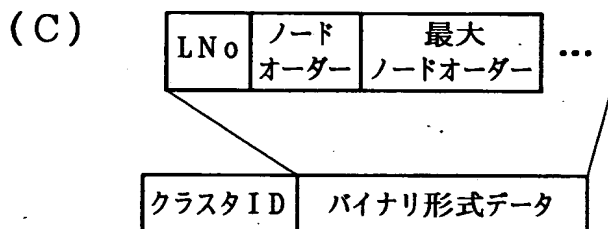
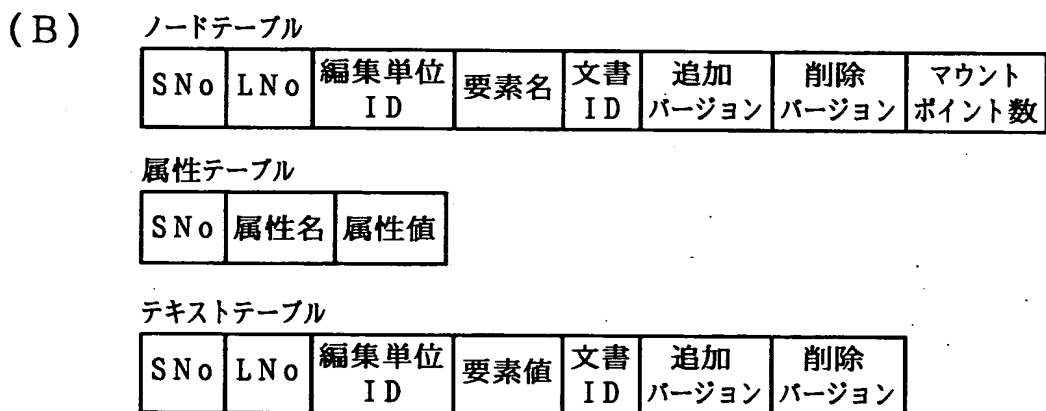
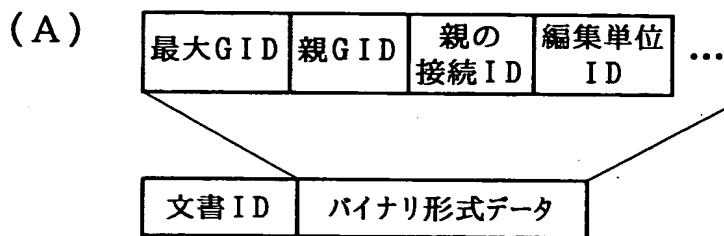
GID = 2

SNo	LNo	要素名	属性値
8	0	内容	...
9	1	文章	...
10	2	段落1	...
11	3	段落2	...
12	4	図表	...

(C)



【図 4】



【図 5】

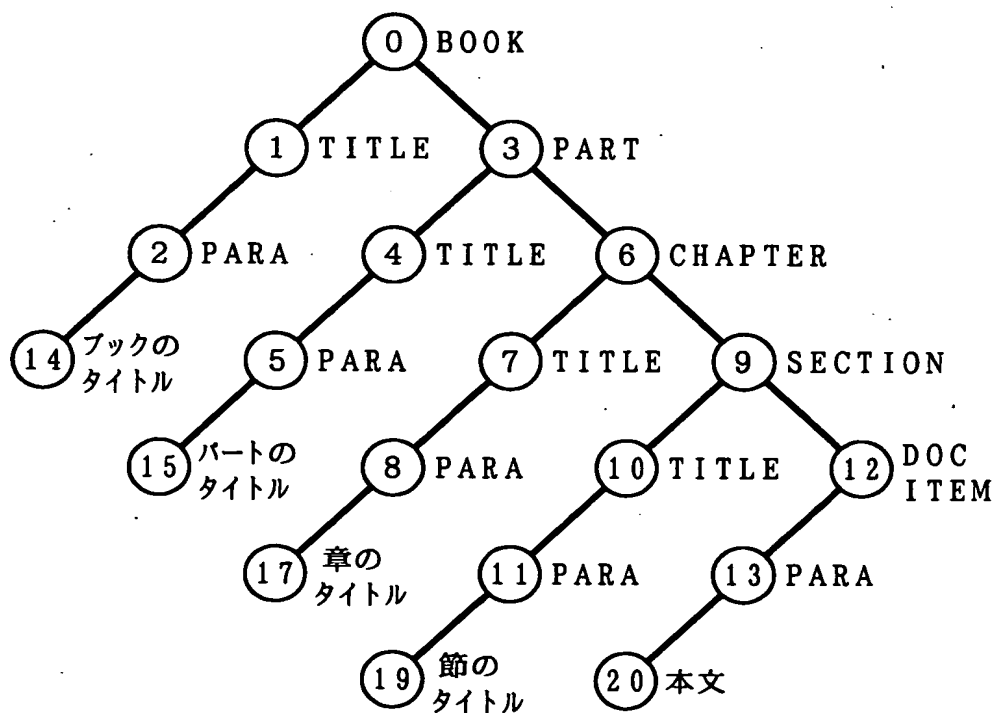
(A)

```

<BOOK ID="F00000000000000">
<TITLE ID="F00000000000001" READING="ぶっくのたいとる">
<PARA ID="F00000000000002">ブックのタイトル</PARA>
</TITLE>
<PART ID="F04000000000000">
<TITLE ID="F04000000000002" READING="ばーとのたいとる">
<PARA ID="F04000000000003">パートのタイトル</PARA>
</TITLE>
<CHAPTER ID="F04010000000000">
<TITLE ID="F04010000000002" READING="しょうのたいとる">
<PARA ID="F04010000000003">章のタイトル</PARA>
</TITLE>
<SECTION ID="F04020000000000">
<TITLE ID="F04020000000002" READING="せつのたいとる">
<PARA ID="F04020000000003">節のタイトル</PARA>
</TITLE>
<DOCITEM ID="F04020000000004">
<PARA ID="F04020000000005">本文</PARA>
</DOCITEM> </SECTION> </CHAPTER> </PART> </BOOK>

```

(B)

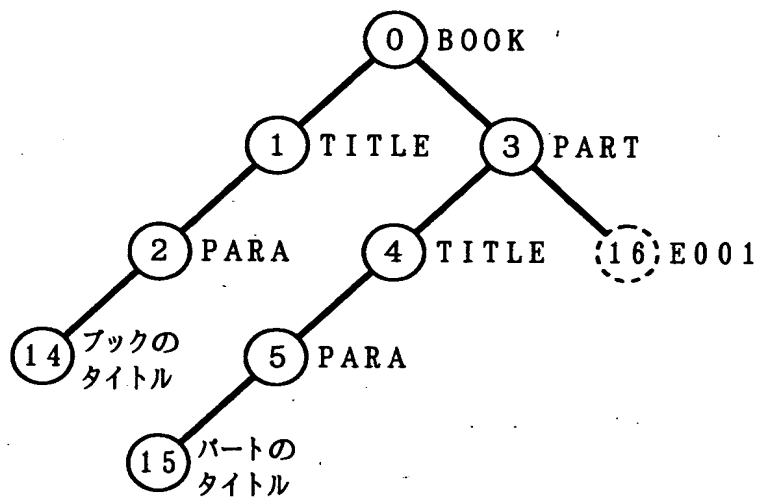


【図 6】

(A) 編集単位 ID=0, GID=0

```
<BOOK ID= "F000000000000000" >
<TITLE ID= "F000000000000001" READING= "ぶっくのたいとる" >
<PARA ID= "F000000000000002" > ブックのタイトル </PARA>
</TITLE>
<PART ID= F040000000000000 >
<TITLE ID= "F040000000000002" READING= "ばーとのたいとる" >
<PARA ID= "F040000000000003" > パートのタイトル </PARA>
</TITLE> &E001; </PART> </BOOK>
```

(B)

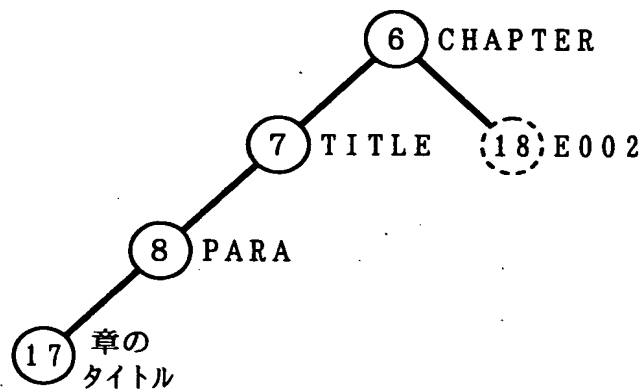


【図 7】

(A) 編集単位 ID=1, GID=1

```
<CHAPTER ID= F040100000000000 >
<TITLE ID= "F040100000000002" READING= "しょうのたいとる">
<PARA ID= "F040100000000003"> 章のタイトル </PARA>
</TITLE> &E002; </CHAPTER>
```

(B)

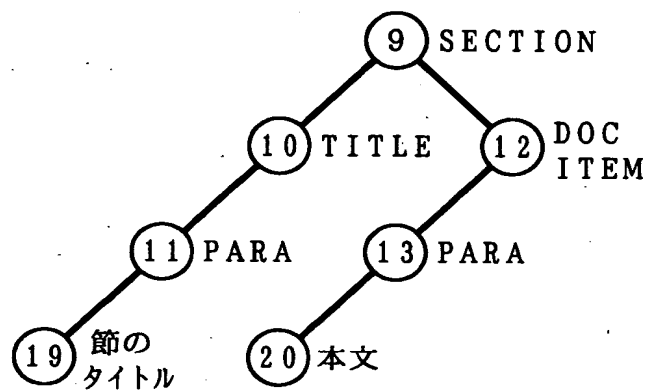


【図 8】

(A) 編集単位 ID=2, GID=2

```
<SECTION ID="F04020000000000">
<TITLE ID="F040200000000002" READING="せつのたいとる">
<PARA ID="F040200000000003">節のタイトル</PARA>
</TITLE>
<DOCITEM ID="F040200000000004">
<PARA ID="F040200000000005">本文</PARA>
</DOCITEM> </SECTION>
```

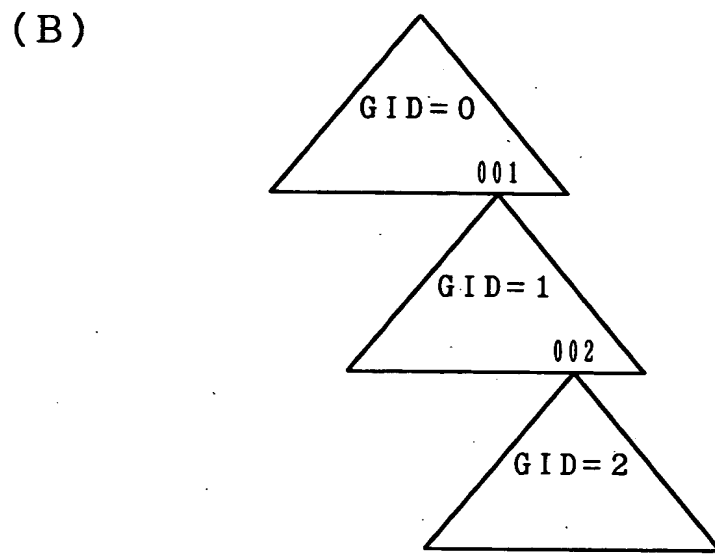
(B)



【図9】

(A)

GID=0				GID=1				GID=2			
2	-1	-1	0	2	0	001	1	2	1	002	2



【図 10】

(A)
ノード
テーブル

SNo	LNo	編集単位 ID	要素名	文書 ID	追加 バージョン	削除 バージョン	マウント ポイント数
0	0	0	BOOK	0	1	NULL	1
1	1	0	TITLE	0	1	NULL	0
2	2	0	PARA	0	1	NULL	0
3	3	0	PART	0	1	NULL	1
4	4	0	TITLE	0	1	NULL	0
5	5	0	PARA	0	1	NULL	0
6	0	1	CHAPTER	0	1	NULL	1
7	1	1	TITLE	0	1	NULL	0
8	2	1	PARA	0	1	NULL	0
9	0	2	SECTION	0	1	NULL	0
10	1	2	TITLE	0	1	NULL	0
11	2	2	PARA	0	1	NULL	0
12	3	2	DOCITEM	0	1	NULL	0
13	4	2	PARA	0	1	NULL	0

(B)
属性
テーブル

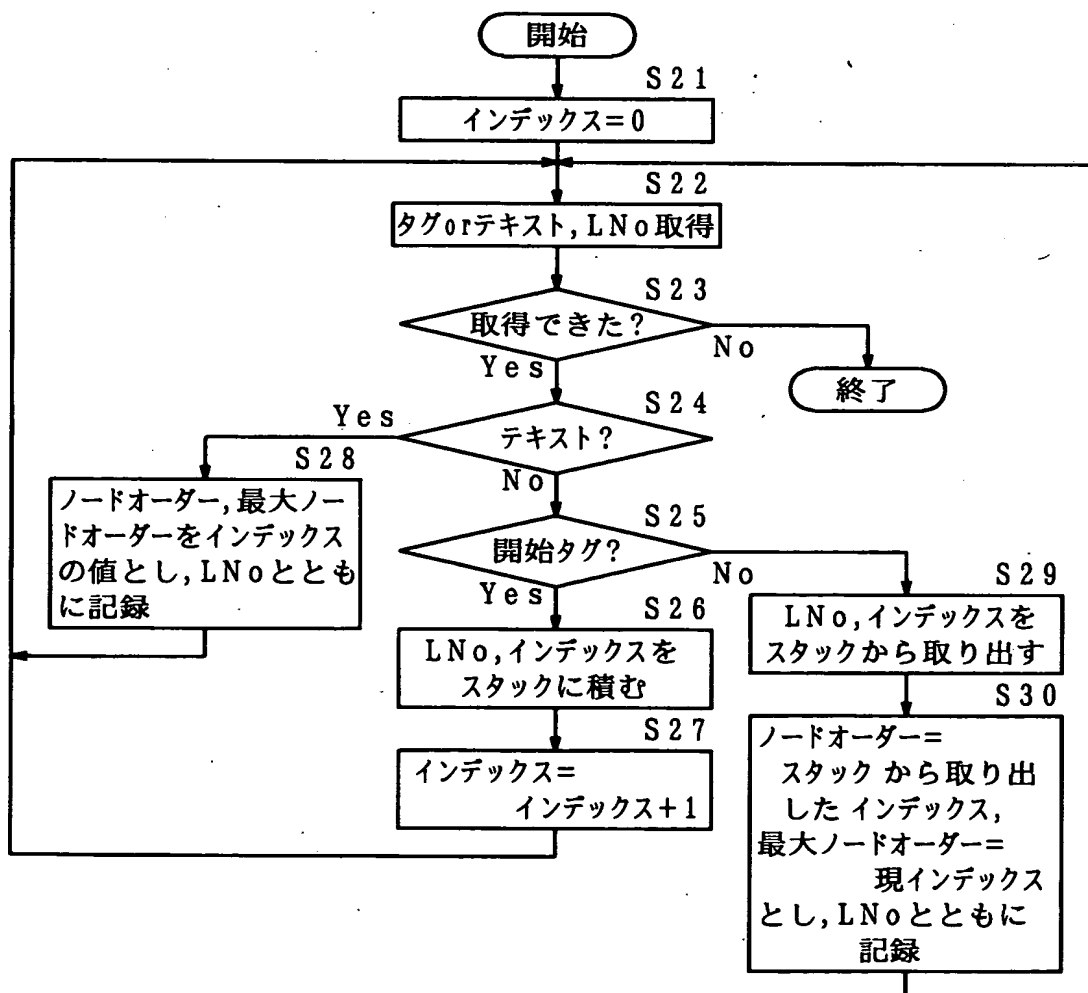
SNo	属性名	属性値
0	ID	F000000000000000
1	ID	F000000000000001
1	READING	ぶっくのたいとる
2	ID	F000000000000002
3	ID	F040000000000000
4	ID	F040000000000002
4	READING	ばーとのたいとる
5	ID	F040000000000003
6	ID	F040100000000000
7	ID	F040100000000002
7	READING	しょうのたいとる
8	ID	F040100000000003
9	ID	F040200000000000
10	ID	F040200000000002
10	READING	せつのたいとる
11	ID	F040200000000003
12	ID	F040200000000004
13	ID	F040200000000005

【図 1 1】

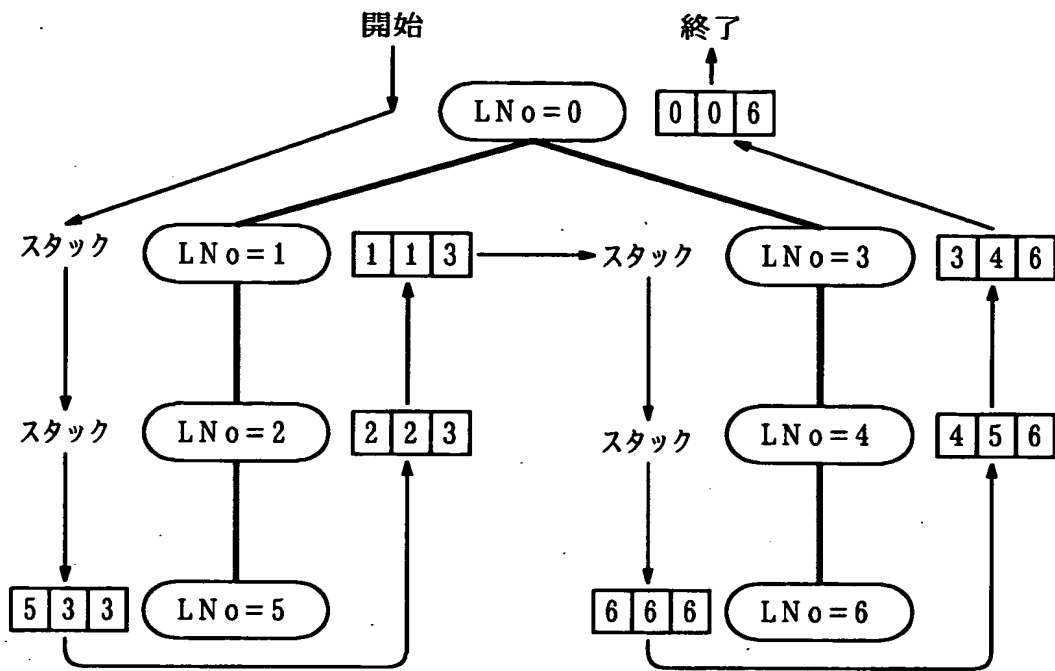
テキストテーブル

SNo	LNo	編集単位 ID	要素値	文書 ID	追加 バージョン	削除 バージョン
14	6	0	ブックのタイトル	0	1	NULL
15	7	0	パートのタイトル	0	1	NULL
16	8	0	E001	0	1	NULL
17	3	1	章のタイトル	0	1	NULL
18	4	1	E002	0	1	NULL
19	5	2	節のタイトル	0	1	NULL
20	6	2	本文	0	1	NULL

【図 1 2】



【図 13】



【図 14】

(A) 編集単位ID=0, クラスID=0

LNo	ノード オーダー	最大 ノードオーダー
0	0	8
1	1	3
2	2	3
3	4	8
4	5	7
5	6	7
6	3	3
7	7	7
8	8	8

(B) 編集単位ID=1, クラスID=1

LNo	ノード オーダー	最大 ノードオーダー
0	0	4
1	1	3
2	2	3
3	3	3
4	4	4

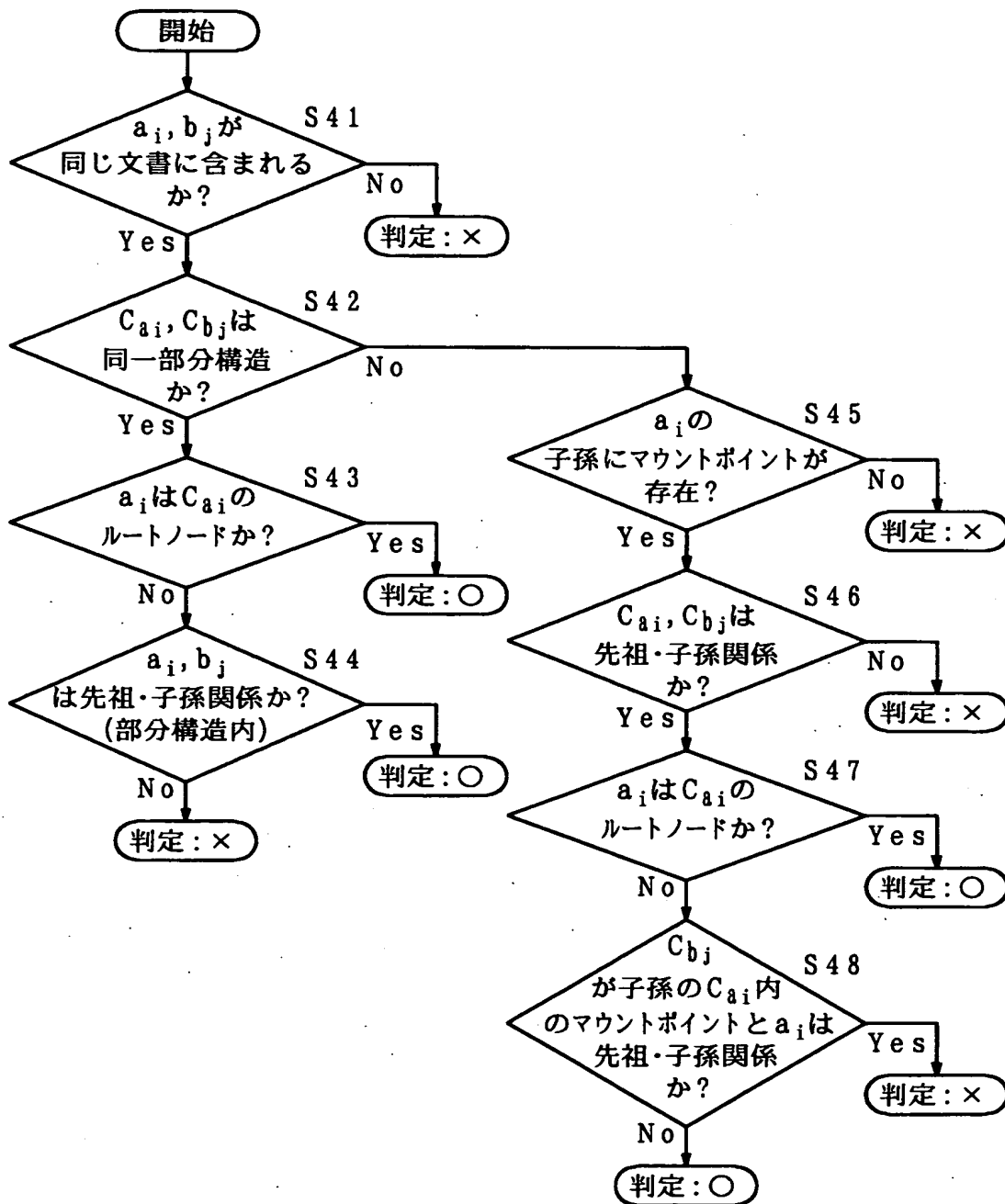
(C) 編集単位ID=2, クラスID=2

LNo	ノード オーダー	最大 ノードオーダー
0	0	6
1	1	3
2	2	3
3	4	6
4	5	6
5	3	3
6	6	6

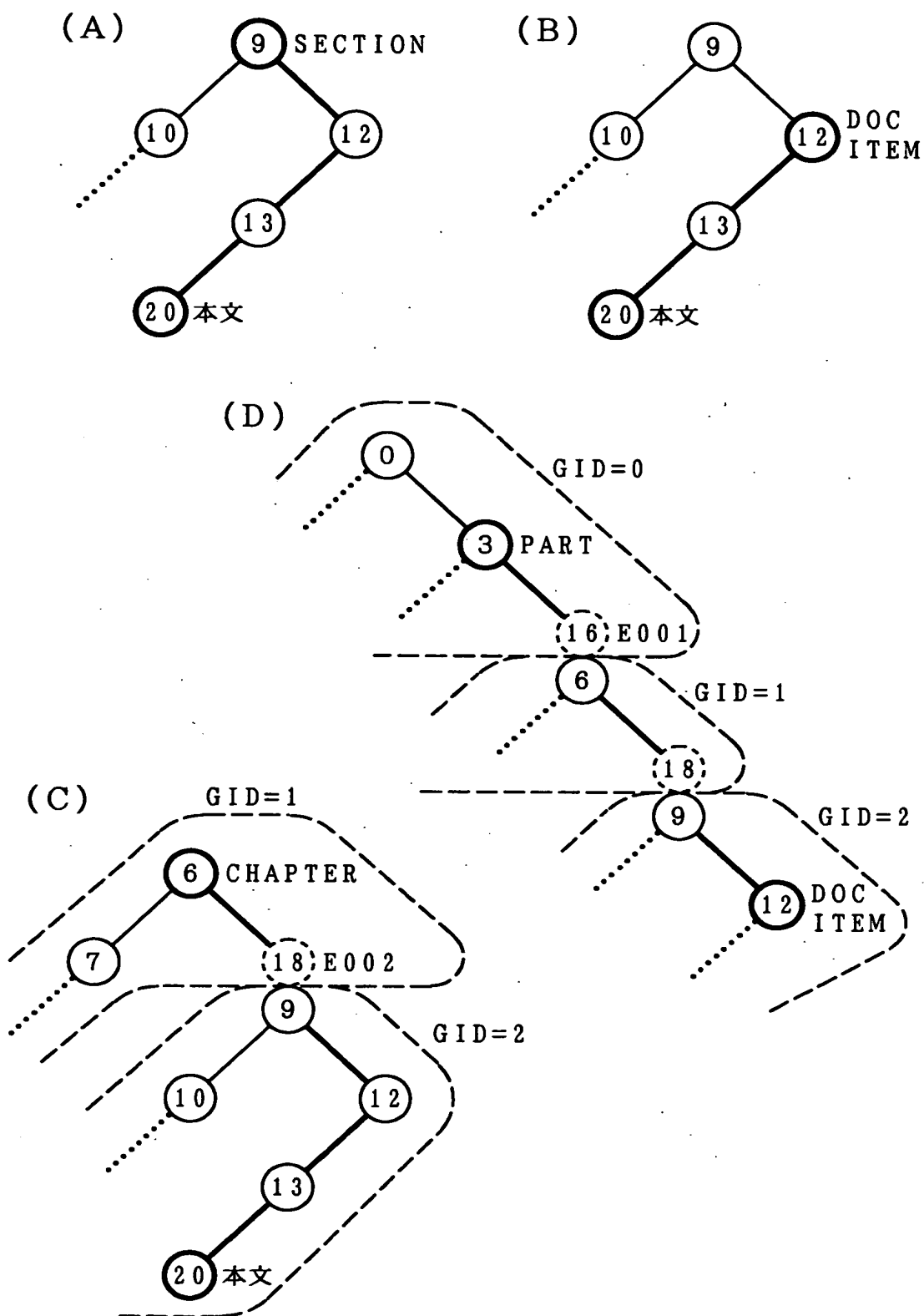
(D) バージョンテーブル

クラス ID	編集単位 ID	バージョン	文書 ID
0	0	1	0
1	1	1	0
2	2	1	0

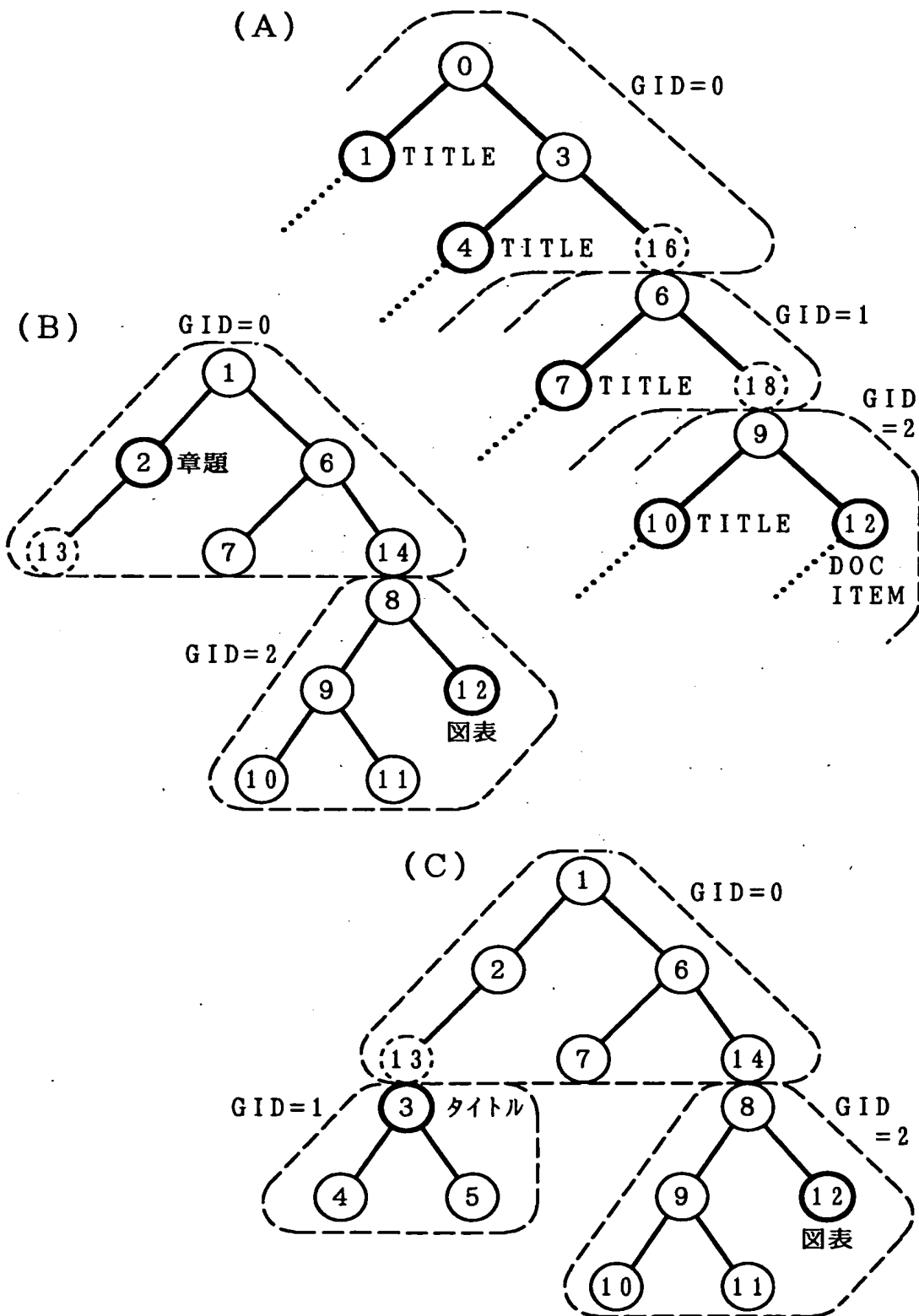
【図 15】



【図16】



【図17】



【書類名】 要約書

【要約】

【課題】 複数人が共同して編集可能な環境を提供できる構造化文書管理装置及び方法と、構造化文書中の要素の先祖・子孫関係を効率よく判定して文書構造の検索を高速に行う検索装置及び検索方法を提供する。

【解決手段】 文書入力部 1 から入力された構造化文書は、文書処理部 5 で編集単位となる部分構造に分解され、部分構造間の関係を示すグローバル構造情報がリレーショナルデータベース 13 に登録される。また、分解された部分構造は編集単位処理部 8 に入力され、各要素の情報が要素情報として、また部分構造中の各要素間の関係が構造情報として、リレーショナルデータベース 13 に登録される。2 要素間の先祖・子孫関係は、グローバル構造情報による部分構造間の先祖・子孫関係と、先祖側の部分構造の構造情報による先祖・子孫関係の判断によって、簡単に、しかも高速に求めることができる。

【選択図】 図 1

出 願 人 履 歴 情 報

識別番号 [000005496]

1. 変更年月日	1996年 5月29日
[変更理由]	住所変更
住 所	東京都港区赤坂二丁目17番22号
氏 名	富士ゼロックス株式会社



Creation date: 06-24-2004
Indexing Officer: TBUI1 - THU-TRANG BUI
Team: OIPEBackFileIndexing
Dossier: 10014661

Legal Date: 06-24-2004

No.	Doccode	Number of pages
1	ECBOX	1

Total number of pages: 1

Remarks:

Order of re-scan issued on